B.E. PROJECT ON

DVESH PRAHARI:
DETECTING VINDICTIVE BEHAVIOUR FROM SOCIAL MEDIA IN CODE SWITCHED LANGUAGES USING NLP AND VISION TECHNIQUES

Submitted By:

Kshitij Rajput (276/CO/15)

Raghav Kapoor (322/CO/15)

Under the Guidance of

Ms. Preeti Kaur

A Project in partial fulfillment of requirement for the award of

B.E. in

Computer Engineering

DIVISION OF COMPUTER ENGINEERING

NETAJI SUBHAS INSTITUTE OF TECHNOLOGY

UNIVERSITY OF DELHI

2019

CANDIDATES' DECLARATION

The project titled **"Dvesh Prahari: Detecting vindictive behaviour from social media in code switched languages using NLP and vision techniques"** is a record of bonafide work carried out by us in the Department of Computer Engineering, University of Delhi, New Delhi, under the supervision and guidance of Ms. Preeti Kaur in partial fulfilment of requirements for the award of the degree of Bachelor of Engineering in Computer Engineering, University of Delhi in the academic year 2019.

The results presented in this thesis have not been submitted to any other university in any form for the award of any other degree.

Dated :

Kshitij Rajput — 276/CO/15

Raghav Kapoor — 322/CO/15

ACKNOWLEDGEMENT

This project's success is directly attributed to the assistance and support from all of those individuals involved in this five month period from January 2019 - May 2019.

We would like to express our sincere gratitude towards our mentor, Ms. Preeti Kaur (Associate Professor, Computer Engineering Department), Netaji Subhas Institute of Technology, Delhi, under whose supervision we completed our work. Her invaluable suggestions, enlightening comments and constructive criticism always kept our spirits up during our work. She was always available to help whenever we faced any problems.

Our experience in working together has been great and fruitful. We learnt a lot in the process. The knowledge, practical and theoretical, that we have gained through this project will help us in our future endeavors in the field. We also learnt the importance of working as a team effectively and efficiently.
We are also grateful to our friends who provided critical feedback and support whenever required.

Above all, I thank Almighty God for giving us this opportunity, being with us and guiding us in all situations and making our way through each and every problem.

नेताजी सुभाष प्रौद्योगिकी संस्थान
NETAJI SUBHAS INSTITUTE OF TECHNOLOGY
(An Institution of Govt. of NCT of Delhi-Formerly, Delhi Institute of Technology)
Azad Hind Fauj Marg, Sector -3, Dwarka, New Delhi-110078
Tel. : 25099050, Fax : 25099025, Website : http://www.nsit.ac.in

CERTIFICATE

This is to certify that the report entitled **"Dvesh Prahari: Detecting vindictive behaviour from social media in code switched languages using NLP and vision techniques"** being submitted by Kshitij Rajput (276/CO/15), Raghav Kapoor (322/CO/15) to the Department of Computer Engineering, NSIT, for the award of bachelor's degree of engineering, is the record of the bonafide work carried out by them under our supervision and guidance. The results contained in this report have not been submitted either in part or in full to any other university or institute for the award of any degree or diploma.

Supervisors:

Ms. Preeti Kaur,

Associate Professor

Department of COE

Abstract

With the uprise of social media, the increase within the variety of social media users has upsurged the hateful content being uploaded on-line. In countries like India, wherever multiple languages are spoken, these abhorrent posts aren't solely from a selected language, however rather from a peculiar mix of languages referred to as code-switched languages. In India, a selected mix, namely Hinglish is the foremost and widespread. Task of hate speech detection gains its complexity from the very fact that there isn't any fastened spellings, synchronic linguistics and linguistics for this language. Also, this hate speech is pictured with the assistance of pictures to form “Memes" which produce a protracted lasting impact on the human mind. This poses a considerable threat for the users that usually become a victim of the disparaging speech and abuses on such platforms. In this project, we have taken up the task of hate and offence detection from text as well as multimodal knowledge, i.e. pictures (Memes) that contain text in code-switched languages. We work upon a memes dataset called Indian political Memes (IPM) dataset and hinglish tweets dataset. We first analyze the different machine learning models used for text classification to know the best applicable for this task. We also work upon transfer learning and propose an LSTM based model. We have also studied and suggested a binary-channeled CNN and LSTM based model in which we separately method the pictures using CNN model and text extracted from the pictures by LSTM model and eventually recombine them to urge the result. This model outperforms all other models to generate state of the art leads to the domain of offensive image (Memes) classification in code-switched languages - Hinglish.

List of Tables

List of Figures

List of Abbreviations

NLP: Natural Language Processing

CLIR: Cross Language Information Retrieval

POS: Part of Speech

LIDF: Language Identification Model

LDA: Latent Dirichlet Allocation

NER: Named Entity Recognition

IPM: Indian Political Memes

MI: Multilingual Index

CNN: Convolution Neural Network

LSTM: Long Short Term Memory

RF: Random Forest

DT: Decision Tree

SVM: Support Vector Machine

MED: Minimum Edit Distance

CBOW: Common Bag of Words

GloVe: Global vectors for Word Representation

GLCM: Gray-Level Co-occurrence Matrix

Contents

1. Introduction

1.1 Hate speech Problem on Social Media

The spark of Natural Language Processing commonly known as NLP began around 60-70 years before, wherein the researchers worked upon large amounts of language data focusing on the different aspects of the natural or spoken language, analyzing their semantics, features and other prominent attributes and determining how the machines interact with these human languages. Also, the growth of internet in the last 25-30 years has led to collection of huge amount of data from online sources and has given rise to a completely new domain of research called data mining. Various tasks have been taken up by the researchers in the past few years in these domains like sentiment analysis, POS tagging, machine translation and many more. However, with the advent of internet and increasing popularity of social media among the masses, one of the most challenging problems in the field of Natural Language Processing that has cropped up in the past few years is that of hate speech detection from social media content.

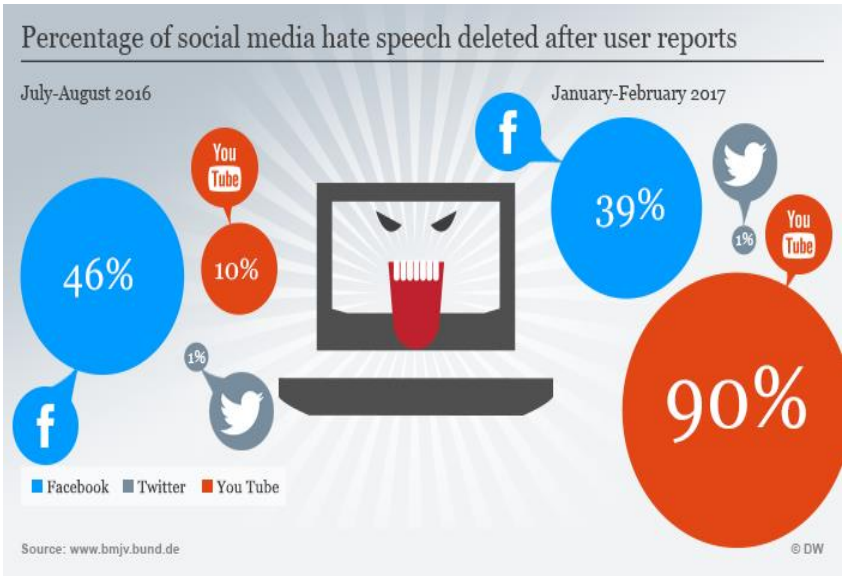


Figure 1 Percentage of social media hate speech deleted after user reports

As the various social media platforms like Facebook and Twitter became famous among the users, they started expressing their feelings and views more freely in real time. This led to the problem of hate inducing and abusive posts on the internet. People would exert their anger on a government policy or somebody's views by writing a hateful or an abusive post against that person. Though the right to show dissent on any view is part of

this hate is expressed in Memes by the democracy but venting out the anger in the form of a hateful or an abusive post online is not the correct way of expressing dissent. Moreover as a recent trend we see that certain people use social media as a platform to instill hate in the minds of the users against a particular religion or gender. Also at the time of elections it is to be seen that many new Facebook pages and Twitter handles crop up whose main aim is to write abusive and hate inducing posts against the opponents which are mostly fake.

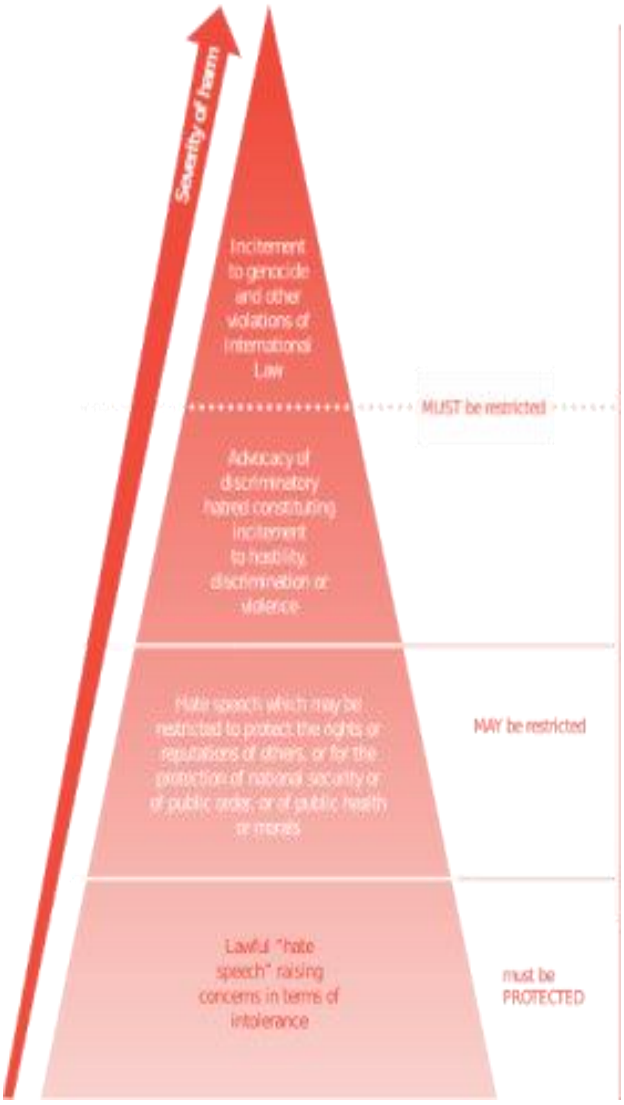


Figure 2 Hate speech Pyramid

Some of the primary purposes for which such content is posted online is personal branding, humor (*Memes*), digital marketing, political canvassing. A large proportion of these images give rise to the problem of hate speech. With growing popularity of social media; hate inducing and violent content is growing humongous and this is specifically during the time of elections or any such political event. In many cases this hate is expressed in Memes by the means of sarcasm. Users often club these images with text to convey their angst. Hate speech detection in such imageries is an intricate task as it involves the analysis of images as well as of the text within it.

The flooding of social media by hateful posts is not good if we want the social media to be clean and fit to use for children and women. Hence the task of hate speech detection becomes a very important task in the domain of Natural Language Processing (NLP).

1.2 Code Switched Languages

Though some work has been done in the field of hate speech detection in English and other languages like Hindi and Bengali separately, very little work has been done in the code switched and the code mixed version of the two languages. The alternation of languages across sentence boundaries is known as code-switching. For example, “Mausam pyaara hai. Let's go to play!” are two different sentences, one in Hindi and the other in English, which translates to “The weather is great. Let's go to play!” is a code switched version of Hindi and English (Hinglish). The alternation within a sentence is known as code-mixing. For example, “Mai bahut khush hun because I finally finished the BTP today” meaning, “I am happy because I finished the BTP today” is a text of Hindi-English code mixed language known as Hinglish.

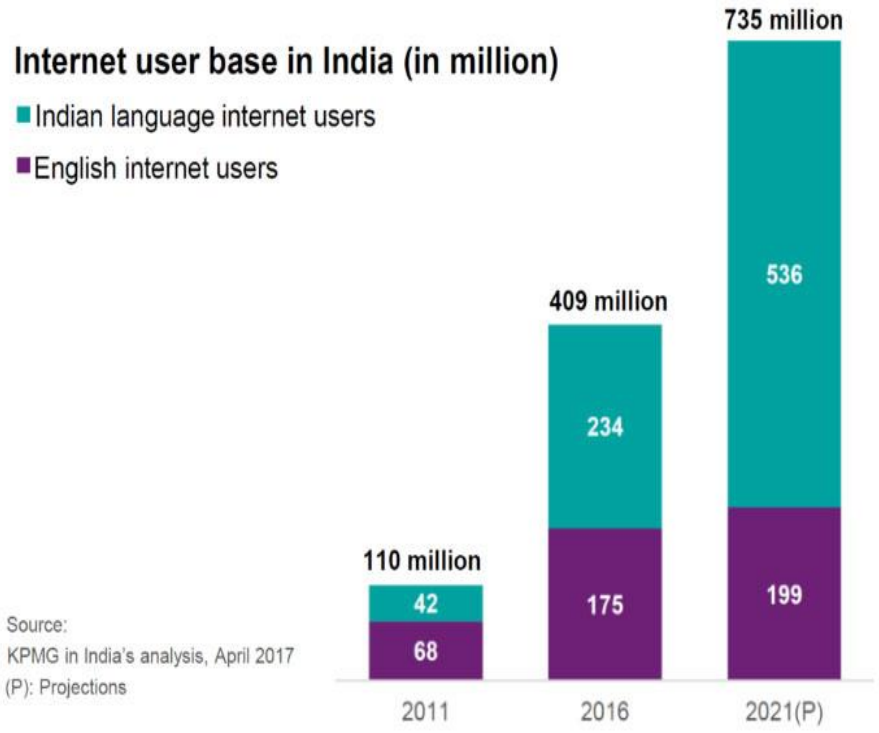


Figure 3 Internet user base in India

According to the ICUBE 2018 report whose main task is to track digital adoption and usage trends in India, it was noted that the number of internet users in India has registered an annual growth of 18 percent and is estimated at 566 million as of December 2018, a 40 percent overall internet penetration among the population. Since most of the users in India know more than two languages, they generally use a code switched version of two

languages in order to express their feelings on social media. This means that the major chunk of data that is produced on daily basis by the social media posts is of code switched languages. Hence if we want to free the social media from hateful posts in order to make it a clean environment for everyone, we should consider hate speech detection in code switched languages. A code switched language that has gained popularity among the users on social media is Hinglish (code switched version of Hindi and English) used mainly in the northern part of India. The Figure 4 shows percentage of Hindi speakers in India as per 2018.

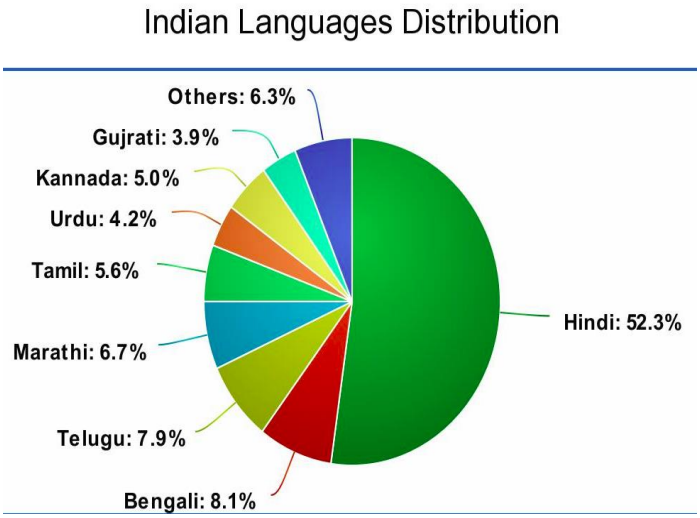


Figure 4 Indian Language Distribution

Hate speech detection in Indian languages is a complex problem due to its rich linguistic diversity [1]. The world of social media has also given rise to the code switched languages. In India, a particular pair of code mixed language, Hinglish is most popularly used. The task of hate speech detection in a code mixed or a code switched language is not as straightforward as it is for a monolingual language. Code Switched languages consists of non-fixed grammar, irregular semantics and spellings. The plethora of slangs and profane words and randomized spelling variations is one of the few characteristics which make the task of hate speech detection much difficult in a code-switched language than for ordinary languages. In Hinglish language, the words are written in *Roman* script instead of the *Devnagari* script, however the meaning of the words are those that in Hindi. The same word can be written in many ways. For example, in the sentence "ye bahut swaad hai", the word *swaad*, which means tasty, can be written as *swad*, *swaad*, *svaad*, *svad* which all mean the same thing. Also, the word to word translation of the sentence would be "This very tasty is" which is grammatically incorrect in English. Hence, as we see, the task gains even more complexity when exhibited in code switched languages. The vast number of social media users, the rise in hate speech, ambiguities in

the semantics and grammar of Hinglish, labyrinth of image analysis is what demonstrates the magnitude of the problem.

1.3 Natural Language Processing (NLP)

Natural Language processing (NLP) is a part of analysis and application that explores how computers perceive and manipulate language text or speech to try and do helpful things. Information science researchers aim to collect data on how people in general perceive and use language so that applicable tools and techniques can be developed to form computer systems that can perceive and manipulate natural languages to perform the required tasks. The foundations of information science dwell variety of disciplines like computer and data sciences, linguistics, arithmetic, electrical and electronic engineering, computing and artificial intelligence, psychology, etc. There are numerous applications of Natural Language Processing, namely natural language text processing, text summarization, machine translation, image captioning, artificial intelligence and expert systems, multilingual and cross language information retrieval (CLIR), speech recognition, , and so on. Basic architecture of NLP system is shown in Figure 5.

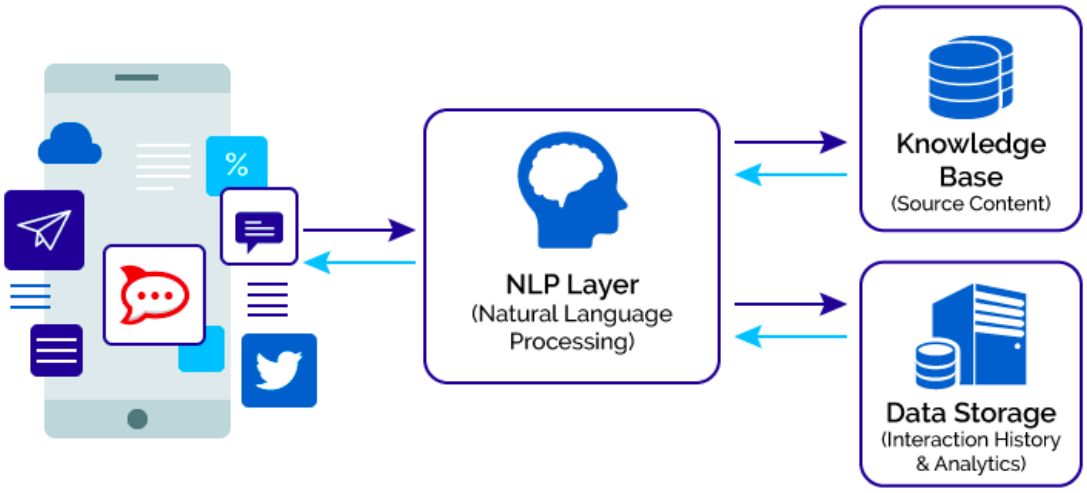


Figure 5 Basic Architecture of NLP system

At the heart of any Natural Language Processing (NLP) task there is a necessary issue of natural language understanding. There are three major problems in developing computer programs that understand natural language, which are: (i) relating to the thought process,

(ii) knowledge of the world, and (iii) illustration and significance of the linguistic input. Thus, an NLP system may begin at the word level n to determine the morphological structure, nature (such as part-of speech, meaning) etc. of the word n and then may move on to the sentence level to determine the word order, grammar, meaning of the entire sentence, etc. A given word or a sentence can have a particular meaning or connotation in a given context and might be associated with several other alternative words and/or sentences within the given context.

Figure 6 Natural Language Processing (NLP) pyramid

In order to comprehend natural languages, it's necessary to be ready to distinguish among the subsequent seven independent levels that people use to extract the meanings out of text or spoken languages: These seven independent levels are:

- Phonetic or synchronic linguistics level that deals with pronunciation
- Levels of morphology dealing with the atomic parts of words, which provide a meaning to the sentence.
- Lexical level that deals with significance of words and elements of speech analyses.
- Grammar level that deals with synchronic linguistics and structure of sentences.
- Linguistic level dealing with the significance of words and sentences
- Pragmatic level dealing with the information that comes from the outer world, i.e., from out of the document.

All these seven independent levels of analysis may be involved in a natural language processing system. Figure 6 shows a Natural Language Processing (NLP) pyramid.

1.4 Supervised Machine Learning

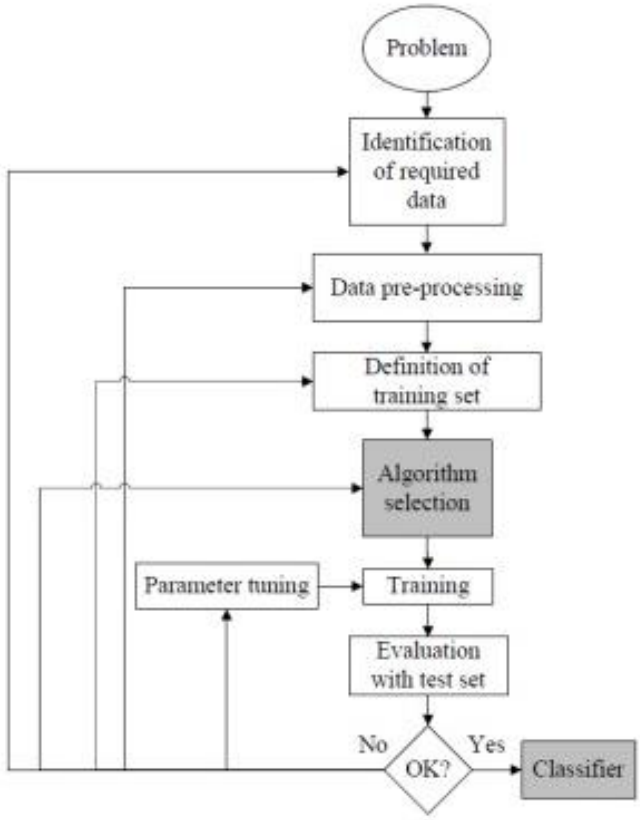


Figure 7 Supervised Machine Learning pipeline

Supervised machine learning is defined as the hunt for algorithms that reason from outwardly equipped instances to provide general hypotheses that then create predictions concerning future instances. Simply, the aim of supervised machine learning algorithm is to produce an abridged model of the distribution of labels in terms of predictor options. The architecture of a supervised machine learning pipeline is shown in Figure 7. The resulting classifier from the supervised machine learning algorithm is then used to assign class labels to the testing instances where the values of the predictor features are known. The differences between supervised and unsupervised learning algorithm are shown in the Figure 8.

Data mining is an important application of machine learning. People are typically vulnerable to creating mistakes in analysis or, possibly, once attempting to determine relationships between multiple features. This makes it hard for them to search out solutions to some issues.

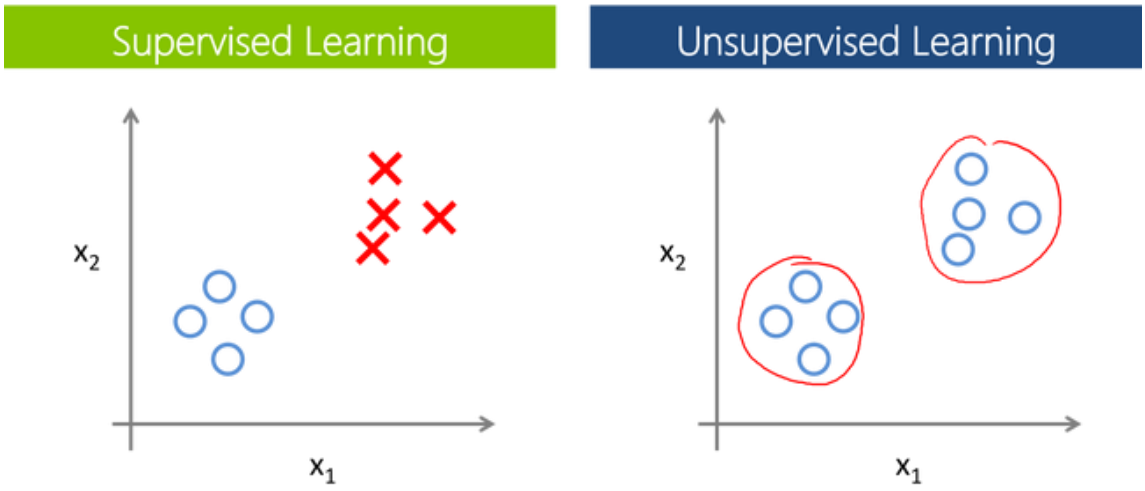


Figure 8 Supervised vs. Unsupervised Learning

These problems can be solved effectively using machine learning algorithms, improving the efficiency of systems and the designs of machines. All the instances in a dataset which is used by a machine learning algorithm are represented using some set of features. These features may be continuous, binary or categorical. If the labels of the instances are known, then the learning algorithm is a supervised one, whereas if the instances are unlabeled it is called a unsupervised machine learning algorithm. By the help of these unsupervised (clustering) machine learning algorithms, researchers aim to find unknown, but useful classes of items. Numerous Machine learning applications have jobs that can be fulfilled using supervised machine learning algorithms.

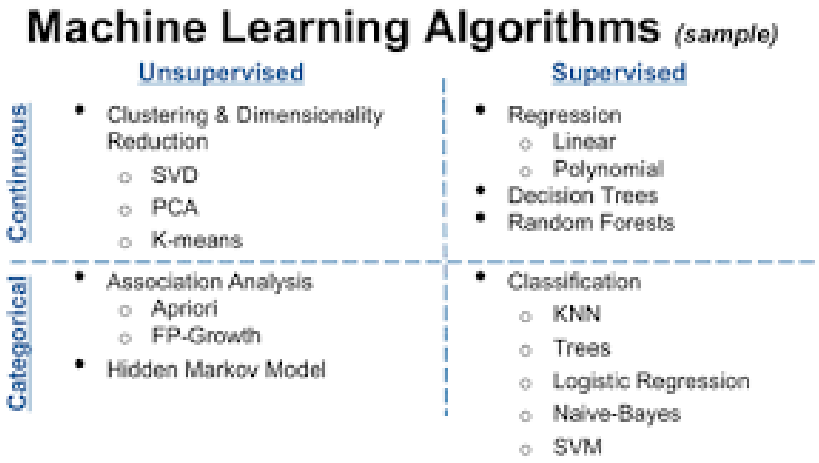


Figure 9 Sample Machine Learning Algorithms

1.5 Multimodal Analysis

Multimodal analysis includes the analysis of communication in all of its forms, however is especially involved with texts that contain the interaction and integration of two or more additional semiotic resources, or means of communication, so as to fulfill the communicative functions of the text. Such resources consist of aspects of speech like intonation and other vocal characteristics, the semiotic action of other bodily resources like gesture (hand, face and body), and also the products of human technology such as image and audio recording, painting, drawing, writing, architecture, and in more modern times, interactive computing resources (software and digital media).

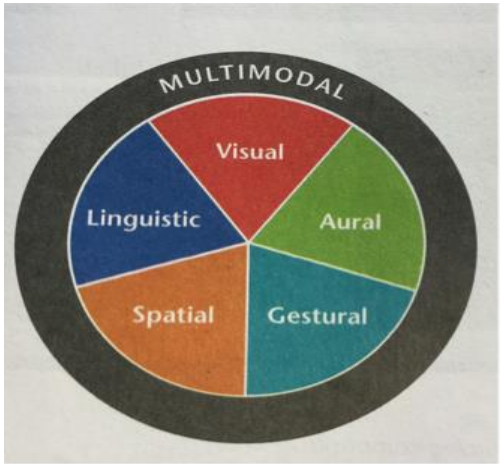


Figure 10 Multimodality

Multimodal discourse analysis defines various ways of approach for learning how social actors develop meaning and how different social actors communicate with other social actors and their surroundings. The social response between an image and its viewer is strongly influenced by whether the subject in the image has an eye contact with the viewer or not. Each of the possibility can be chosen as an example of *mood* where the eye contact could suggest a demand, whereas no eye contact could suggest an offer.

Figure 11 provides a detailed overview of the different parameters of multimodal analysis. Scientifically there are four main semiotic modes can be used to communicate in discourse genres, namely (i) Language (ii)Visual communication (iii) Gesture (iv) Action.

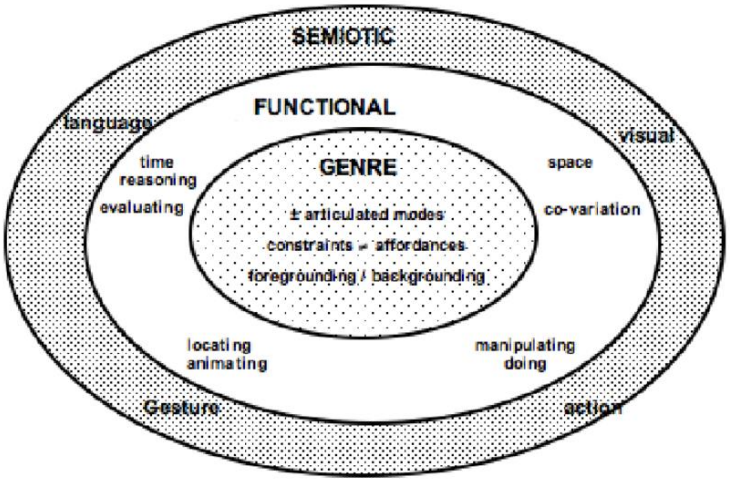


Figure 11 Parameters for multimodal analysis

Every semiotic mode defined above has distinct functional specialisms. For example, natural language whether speech or text, is suitable for the expression of time relations, whereas the visual mode is better suited for articulating spatial relations. The inner circle of Figure 11 indicates the other factors that have to be considered in a multimodal approach to a genre.

2. Background and Related Work

In this section we discuss the various researches that have taken place in the domain of Natural Language Processing (NLP) on code switched data. Here, we will study about the previous work focusing on language identification, POS tagging, named entity recognition, sentiment analysis and hate speech detection in code switched languages. The primary emphasis will be on the work that involves the use of Hindi-English code mixed data.

2.1 Language Identification on Code Switched Language

Language identification is one of the initial tasks of the Natural Language Processing (NLP). The task was performed at first on code mixed Hinglish data by [2] through his research which discussed word level language detection through N-gram Language Profiling and Pruning, Dictionary-Based Detection and SVM-based Word-Language Detection. In SVM based word language detection the task of identifying language is seen as a classification problem. The SVM classifier considered the following features:

(i) N-gram with weights: N-gram was implemented using the bag of words principle.

(ii) Dictionary-based features: This is a binary feature for each language.

(iii) MED-based weight: This is Minimum Edit distance which is calculated if a word is not found in any of the dictionaries.

(iv) **Word context information:** A window of length 7 to determine the contextual information is used as the feature.

The SVM model which considered all the above features was reported to produce a precision score of 90.84 percent for the language identification task. Post processing, the model was shown to produce an even higher precision score of 94.84 percent.

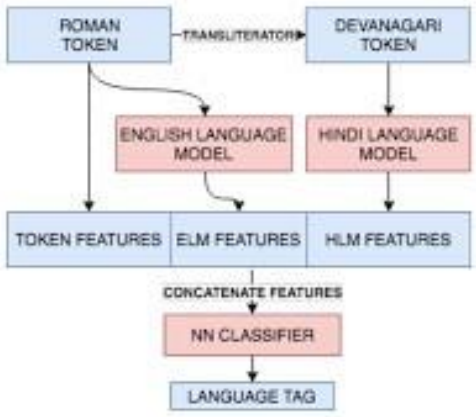


Figure 12 Architecture of LIDF

Mave et al. [3], through his research compared CRF model, Bidirectional LSTM model and a Word character LSTM model for the task of language identification. For the Hi-En code mixed data, the CRF model with context window size of 2 was shown to produce a F1 weighted score of 96.84 percent.

Singh et al. [4], produced a language identification model (LIDF) based on the hypothesis that word or character sequences of different languages encode different structure. Hence the aim was to capture the structure of the character for all the languages which is done by training a model for a token length n, which learns to predict the character at n^{th} position given all the (n-1) characters. The model consisted of a RNN layer each containing 128 LSTM cells with *Relu* activation. The output of the second RNN layer at the last time step is connected to a fully connected (FC) layer with softmax activation. The architecture of the model is shown in Figure 12.

2.2 POS Tagging

Part of Speech (POS) also known as POS tagging, or PoS tagging ,or POST tagging is described as grammatical tagging or word-category disambiguation, i.e. labeling each word in the database as a specific part of speech, depending on definition and context of the word. POS tagging is one of the fundamental pre-processing steps for NLP which has various use cases like generating parse trees and named entity recognition (NERS).

The POS tagging of Hi-En code mixed data was first provided by [5]. The corpus used for the task was created by extracting comments out of the facebook posts of some famous personalities of India like Narendra Modi, Shahrukh Khan, Amitabh Bachchan

and BBC hindi news page. Every sentence was annotated using the following annotation scheme:

(i) Matrix

(ii) Word Origin

(iii) Normalisation/ Transliteration

(iv) Part Of Speech

The example of the annotation scheme is shown in Figure 13. The system proposed by the authors tried to apply a pipelined approach in which the tasks of language identification, normalization and then POS tagging were done in a sequential order. The basic idea behind the POS tagging of code mixed data used was to divide the text into continuous maximum chunks of words whose language are the same. Then Hindi POS tagger was operated on the Hindi chunks and the English POS tagger was operated on the English chunks.

```
<s>
      <matrix name="Hindi">
      love_NOUN/E affection_NOUN/E lekar_VERB=”ले कर” salose_NOUN=सालों
se_ADP=से sunday_NOUN/e ke_ADP=के din_NOUN=दिन chali_VERB=चली aarahi_VERB=”आ
रही” divine_ADJ/e parampara_NOUN=परंपरा ko_ADP=को age_NOUN=आगे badhha_VERB=बढा
rahe_VERB=रहे ho_VERB=हो
      </matrix>
</s>

<s>
      <matrix name="Hindi">
      jindagi_NOUN=ज़िंदगी kaise_PRON=कैसी h_VERB=है paheli_NOUN=पहेली
haye_PRT=हाये
      </matrix>
      <matrix name="English">
      may_ADP his_PRON sol_NOUN=soul rest_VERB in_ADP peace_NOUN
      </matrix>
</s>
```

Figure 13 Example of annotation scheme for POS Tagging

Hence, this task used language identification in order to separate the chunks of data on the basis of their language. CRF++ based POS tagger was used for Hindi text and Twitter POS tagger was used for the English text. This being the initial model, the maximum chunk accuracy reported was 34% when the language identities and normalized forms

were known. For the case where both of these factors were unknown, the maximum chunk accuracy reported was 25%.

Another approach for POS tagging was proposed by [6] which produced much better results than the previous approaches. The datasets used in the experiment were provided by the organizer of POS tagging tool contest at ICON-2016. Nine exhaustive set of features were used for the task of POS tagging. These features were Context word, Character n-gram, Word normalization, Prefix and suffix, Word class feature, Word position, Number of upper case characters, Word probability.

These feature set were used to build a POS model. Conditional random field (CRF) is used was the underlying classifier. CRF ++3, an implementation of CRF was used to perform the experiment. The paper reported a high precision score of 0.782 for the Hi-English code mixed data extracted from twitter. This was a major improvement from the previous approaches.

2.3 Named Entity Recognition

Named Entity Recognition (NER) is the task of labeling all the entities in the database into categories such as organization, states, dates, person, electronics, government etc. Named Entity recognition has several use cases in the field of Natural Language Processing (NLP) like classifying contents from news providers, efficient search algorithms and customer support. The few researches that have taken place for named entity recognition are described below.

The initial work on named entity recognition on code mixed Hi-En data was provided by [7]. The dataset was scraped using the twitter API and the tag set was chosen which the Government of India standardized tag set was. In this tag set, named entity hierarchy was divided into three major classes, i.e. Entity Name, Time and Numerical expressions. The Name hierarchy had eleven attributes, Numerical Expression had four attributes and Time had three attributes. The best results for this task were provided by the team Irshad-IIIT-Hyd as reported by the paper. This team used a simple feed forward neural network with activation function as rectifier, learning rate as 0.03, dropout as 0.5 and learning rule as *adagrad* with L2 regularization. English wiki corpus was used for developing word-embeddings using Gensim Word2Vec. The team was able to secure a precision accuracy of 80.92%.

Other works have been performed for named entity recognition such as the [4], which considers that the named entities can be identified using the features extracted from the words surrounding it. The following features were used for the task: (i) Token based

features (ii) Prefixes and Suffixes (iii) Character based features (iv) Language based features (v) Syntactic features (vi) Tweet capitalization features.

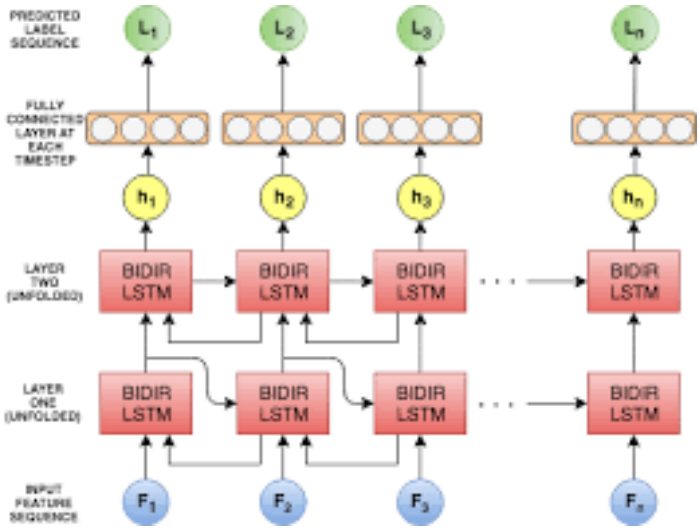


Figure 14 LSTM NER architecture

A LSTM based model comprising of two bidirectional RNN layers using LSTM cells and *Relu* activation was used for the task of using the above features for named entity recognition. The architecture of the LSTM model is shown in Figure 14. Also a standard CRF model proposed by [8] added with a L1 and L2 regularization to prevent over-fitting, was also tested for the task. The CRF model was shown to have the highest precision accuracy of 84.95%, a major improvement from the previous work.

2.4 Sentiment Analysis

Sentiment analysis is the task of labeling the text in the database as positive, negative or neutral. Sentiment analysis on code switched data from internet has great importance in mining of thoughts like analyzing social media propaganda and fake news during elections in multilingual societies. The task of sentiment analysis in the code mixed data of Hindi-English was restrained due to the lack of a suitable annotated dataset.

The task of sentiment analysis on code mixed Hi-En (Hinglish) social media content was first performed by [9] who used sub-word level representations in LSTM architecture to

perform the task. The dataset was created by crawling the comments from the facebook pages of two of the most famous personalities of India, Salman Khan an Indian actor and Narendra Modi, the prime minister of India at the time. Then manual annotation of the dataset was done by two annotators. All the text in the database was labeled into three categories, i.e. positive, negative or neutral. The size of the dataset was 3879 sentences which consisted of 15% negative, 50% neutral and 35% positive comments owing to the nature of conversations in the selected pages. This research used intermediate sub-word feature representations learned by the filters during convolution operation. The relevant Subword representation was circulated with LSTM using which, the final sentiment of the sentence was calculated. The architecture used and the proposed methodology for sentiment analysis [9] is shown in the following Figure 15.

The subword level representation using LSTM performed better than the system proposed by the [10] on this dataset with a accuracy of 69.7 percent and a F1 score of 0.658. The system also outperformed the [11] which used SVM unigram and unigram + bigram features and also the Lexical Lookup method proposed by [12]. The system performed better than the previous systems not only in the Hi-En code mixed dataset prepared but also on SemEval' 13 Twitter Sentiment analysis dataset with a accuracy of 60.57% and F1 score of 0.537. Hence the subword level LSTM performed significantly better than the baselines.

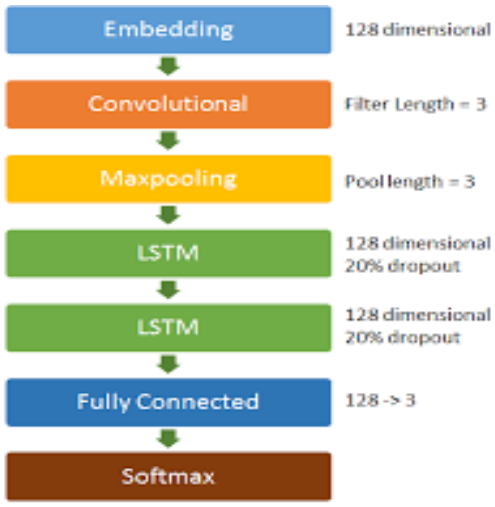


Figure 15 Model architecture for sentiment analysis

Jhanwar et al. [13] proposed an ensemble method for sentiment analysis on Hindi-English code switched dataset. The dataset used in this research was same as the dataset proposed by [9].

The ensemble model combined the outputs of character-trigrams based LSTM model and word N-gram based MNB model to predict the sentiment of Hi-En code-mixed texts. While the LSTM model encoded deep sequential patterns in the text, MNB captured low-level word combinations of keywords to compensate for the grammatical inconsistencies. This ensemble model produced a accuracy of 70.8% and a F1-score of 0.661 which was better than that of [9]. The Fig. 6 shows the architecture of the ensemble classifier used by [13].

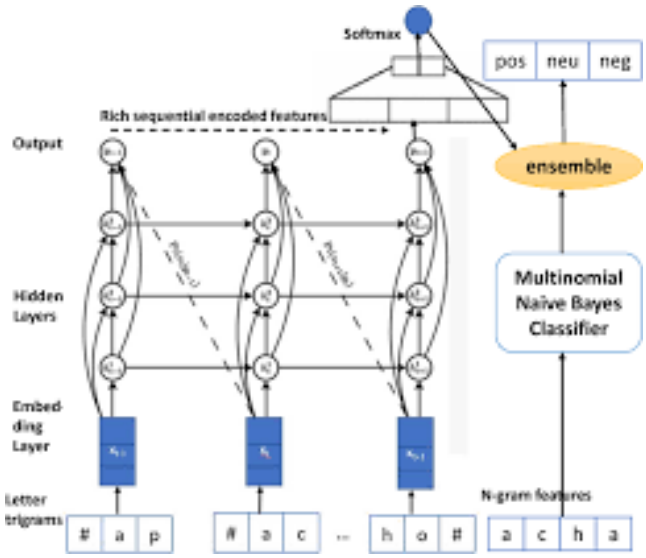


Figure 16 Ensemble classifier for sentiment analysis

Other methods have been proposed by [14] which performs the task of sentiment analysis using CNN which basically consists of a sentence representation matrix, convolution layer, pooling layer and fully connected layer. The system was able to provide a high precision score 70.25% which outperformed the baseline models which used cosine similarities and SVM for the task of sentiment analysis.

2.5 Hate Speech Detection

The task of automatic detection of hate speech is fast becoming an important problem in today's world with the increasing penetration of the internet among the users in the past decade. The initial task for hate speech detection was performed by [15], who developed a prototype system *Smokey* for detecting email flames (angry or offensive emails) using a 47 elements feature set which captured the syntax and semantics of the sentences present in the dataset. These features were passed to a decision tree generator to categorize the emails as flames or not. Yin et al. [16] used *libSVM* as the classifier model with local features, sentiment features and contextual features for detecting harassment on Web 2.0. A SVM based model trained on a corpus of 1,655,131 user comments on Yahoo buzz, combined with valence analysis for detecting personal insults on social news websites was put forward by [17]. Another work, [18] proposed to use Latent Dirichlet Allocation (LDA) for generating topical features which are passed to a logistic regression (LR) classifier for the task of detecting offensive tweets on a twitter corpus. Using proposed a Convolution Neural Network (CNN) for classification on twitter text data into four categories: sexism, racism , both (racism and sexism) and non-hate-speech using the following features: character 4-grams, word2vec vectors to capture semantic information, randomly generated word vectors, and word vectors combined with character n-grams. A [31] dataset of about 25K tweets labeled as hate inducing, offensive or benign was released by who put forward a logistic regression model with L2 regularization for classifying the tweets in one of the three categories.

However most of the research on hate speech detection in the past was restricted to English text only. The task of hate speech detection on Italian language using the following features: (i) morpho-syntactical features, (ii) sentiment polarity and (iii) word embedding lexicons was shown by [19]. Two types of classifiers were considered, first a SVM based classifier and secondly a LSTM based classifier which were compared for hate speech detection on Italian tweets dataset. However the task of hate speech detection on code switched data has its own intricacies of having to deal with non-fixed spellings, grammar and semantics for this language.

Since our work consists of hate speech detection of memes with text in Hinglish language, so we look at some past work for hate speech detection focusing on Hinglish language. The task of hate speech detection on Hindi-English code switched data using a Random Forest (RF) classifier and a Support Vector Machine (SVM) classifier was performed by [20] using the following features: (i) character n-grams, (ii) word n-grams, (iii) punctuations, (iv) negation words and (v) lexicons.

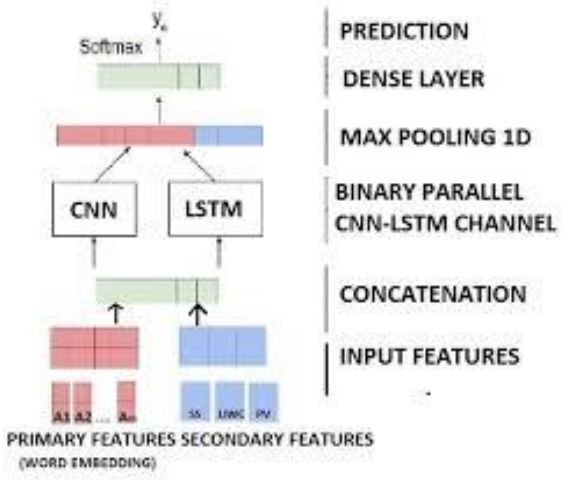


Figure 17 MIMCT Model for hate speech detection

A ternary trans CNN model using transfer learning for hate speech detection on Hindi-English code switched dataset was performed by [21]. Mathur et al. [22] also put forward a MIMCT model which takes in a series of primary and secondary word embeddings into a CNN-LSTM based binary channel neural architecture for hate speech detection. The MIMCT model architecture is shown in the Figure 17.

2.6 Image Analysis

Analysing sentiments from a image is a complex problem in itself and has seen many works in the past. Research by [23] focused on analysing sentiment out of images where the authors proposed a novel mechanism of finding the emotion out of an image by finding an orthogonal three dimensional factor space of a image and then passing it through a SVM classifier. Siersdorfer et al. [24] analysed the relation between sentiment of images expressed in metadata and their visual content in the social photo sharing environment Flickr. Another work by [25] deals with training of a deep convolutional neural network to classify the 1.2 million in ImageNet into 1000 different categories. A progressive CNN model for visual sentiment analysis with transfer learning to learn the features on a twitter image dataset was also proposed by [26]. The progressive CNN model architecture is shown in the Figure 18.

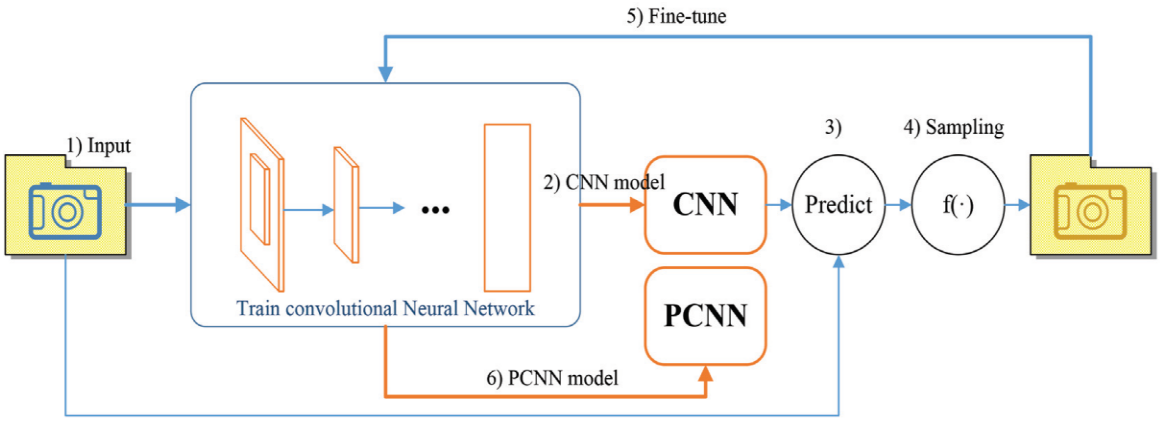


Figure 18 Progressive CNN model for visual sentiment analysis

Since our work focuses on combining both the textual and the image features from a meme, we look at some researches that have performed classification considering such combination of features before. Cai et al. [27] performed sentiment analysis on the combination of text and images instead of considering them separately. Two individual CNN structures were used to capture the image and textual features, which were then passed to another CNN structure to exploit these features and calculate the sentiment. A novel method was proposed by [28] for capturing textual and visual features using deep convolutional networks and then passing these features to a multiple kernel learning classifier for performing the task of multimodal sentiment analysis efficiently. Also, one of the works [29], analyzed sentiment in web videos by building a joint model that takes a combination of audio, visual and textual features.

3. Dataset and Evaluation

3.1 Datasets for Hate Speech Detection

3.1.1 HOT Dataset

HOT dataset is a manually annotated dataset that was used by [22]. This dataset was created by scraping tweets from Twitter during the interval of 4 months, from November, 2017 to February, 2018. Geo-location restrictions were imposed on the tweets so that the tweets hailing only from the Indian subcontinent were made part of the corpus. Since the tweets from famous personalities receive the maximum flak from the users, the tweets and their responses were crawled from the twitter handles of sports personals, political figures, movie stars and news channels. From the initial corpus of 25667 tweets some of tweets containing only URL's, only images and videos, having less than 3 words, non-English and non Hinglish scripts and duplicates were removed.

The annotation was done in three categories that are:

(i) Hate Inducing

(ii) Abusive

(iii) Non offensive

Table 1 Example of tweets and their annotation in HOT dataset

Tweet	Label
(i) Tum ussey pyar kyun nahin karti? (ii) Why don't you love him? (iii) you him love why no	Non-offensive
(i) Ch*d! Yeh sab ch*tiye hain! :/ (ii) F**k! They all are c*nts! :/ (iii) F**k they all c*nts are	Abusive
(i) M*d*rch*d Mus*lm**n sE nafrat (ii) m*therf*ck*r m*sl*m hate (iii) m*therf*ck*r m*s*l*m**n hate	Hate-inducing

The annotation for the dataset was done by three annotators having sufficient knowledge in the domain of NLP. The tweets were annotated as hate inducing if and only if they satisfied one or more of the following conditions: (i) tweet consisted of a sexist or racial barb to malign a minority, (ii) undignified stereotyping or (iii) tweet consist of a hateful hashtag such as #HinduSc*m. The label which was in majority among the three annotators was chosen as the final label. In case of no majority was reached, the final annotation was decided by the help of a NLP expert. Finally there were only 386 tweets

that needed expert annotation. Out of the total 3189 tweets, 2551 tweets were used for training purpose and 638 tweets were used for the testing purpose. An example of some of the tweets from the HOT dataset is shown in Table 1.

Table 2 Distribution of memes in HOT Dataset

Label	**HOT Dataset**
Non Offensive	1121
Abusive	1765
Hate Inducing	303
Total	3189

3.1.2 HEOT Dataset

HEOT dataset was also created by [21] for hate speech detection. HEOT Dataset was created using the Twitter Streaming API by selecting tweets in Hindi-English code switched language which were mined using specific profane words. The dataset was compiled from November, 2017 to December 2017 and was distributed to ten NLP researchers for annotation and verification. The dataset thus created consisted of 3679 tweets which consisted of 1414 non-offensive, 1942 abusive and 323 hate-inducing tweets. The annotation scheme selected for this dataset was similar to that of HOT dataset

(ii) Non offensive

(ii) Abusive

(iii) Hate inducing.

Table 3 Example of tweets in HEOT Dataset

Tweet in HEOT Dataset	Translation in English	Label
Hum sab ghumne jaa rahe hain? http://t...	We all are going outside? http://t...	Non Offensive
@username1 Kutiya! Mujhe mat sikha:/	@username1 B*tch! Do not teach me:/	Abusive
M*d*rch*d aatanki Akbaar ko maara daalo #SaveWorld	M*th*rf*ck*r Kill terrorist Akbaar #SaveWorld	Hate Inducing

Figure 19 Word Cloud for HEOT Dataset

3.1.3 Davidson Dataset

Davidson [31] provided one of the initial datasets for hate speech detection. Though this dataset is in English language and not code switched language, but this dataset is used as the baseline dataset to check the accuracy of the classifying models. This dataset is used by both the primary researches in the field of hate speech detection in code switched language, [30] and [21] to validate the accuracy of their classifying models. The dataset was created by first collecting hate speech lexicons from Hatebase.org and then collecting the tweets that use those lexicons with the help of twitter API. After random sampling of about 25K tweets, each tweet was annotated from three or more users. The final dataset released consisted of 24802 tweets each of which labeled as:

(i) Hate inducing

(ii) Offensive

(iii) Neither.

Figure 20 Word cloud for Davidson Dataset

Table 4 Distribution of tweets in Davidson and HEOT dataset

Label	Davidson Dataset	HEOT Dataset
Non Offensive	7274	1414
Abusive	4836	1942
Hate Inducing	2399	323
Total	14509	3679

3.1.4 IPM Dataset

We constructed a Indian Political Memes (IPM) Dataset for this experiment which consisted of memes that are shared on a day to day basis on internet and are politically motivated. We created this dataset using the *google_images_download* module which is an open source python tool available online, to scrape several images using the keywords as the name of some famous politicians, social activists, journalists and big political events that have taken place post-independence in India. For each keyword 100 images were downloaded using the module, resulting into a corpus of images containing 5000 images. Out of this corpus of images, 1500 memes were randomly sampled and were asked to be annotated by three annotators into the following categories:

(i) Hate Inducing

(ii) Satirical

(iii) Non offensive.

Some of the images that were blurred and consisted of no text were removed from the dataset resulting into a final dataset of 1218 memes. The memes were annotated as hate inducing if and only if the meme satisfied one or more of the following conditions: (i) meme consisted of a sexist or racial barb to malign a minority, (ii) meme had object stereotyping or (iii) meme consists of a hateful hashtag such as #HinduSc*m. The annotators were specifically asked not to consider a meme as hate inducing due to the presence of a particular word, however offensive that word might be. The text from the IPM Dataset was extracted using an OCR (ocr.space) to create a text dataset out of memes. A word cloud is created using the text dataset and is shown in Figure 22.

Table 5 Distibution of memes in IPM Dataset

Label	IPM Dataset
Non Offensive	339
Satirical	452
Hate Inducing	427
Total	1218

Non Offensive Satirical Hate Inducing

Figure 21Example of memes in IPM Dataset

Figure 22 Word cloud for IPM Dataset

3.2 Dataset Evaluation Metrics

3.2.1 Cohen's Kappa Score

The Cohen's Kappa [32] metric is used for determining the inter-annotator agreement between two annotators. The metric determines the quality of annotation by taking into account the possibility that two annotators could have classified the subject into same category by chance. A value of the kappa score close to 0 indicates no agreement between the annotators and a value close to 1 indicates perfect agreement between the two annotators. Mathematically, Cohen Kappa is defined by equation (1).

$$k \equiv \frac{p_0 - p_e}{1 - p_e} \equiv 1 - \frac{1 - p_o}{1 - p_e} \quad (1)$$

Where, p_o denotes the relative observed agreement between the two annotators and p_e denotes the probability of chance agreement between the annotators.

The Cohen's kappa inter rater agreement was calculate for HOT dataset and Davidson dataset as 0.83 and 0.92 respectively. The Cohen's Kappa score for HEOT dataset and IPM dataset are shown in the Table 6 and Table 7 respectively.

Table 6 Cohen's Kappa score for HOT Dataset

	A1	**A2**	**A3**
A1	-	0.76	0.84
A2	0.76	-	0.88
A3	0.84	0.88	-

Table 7 Cohen's Kappa score for IPM Dataset

	A1	**A2**	**A3**
A1	-	0.81	0.77
A2	0.81	-	0.87
A3	0.77	0.87	-

3.2.2 Multilingual Index

The Multilingual Index (MI) is calculated on the text dataset which is prepared by extracting text out of the memes of the IPM dataset. MI is used to calculate inequality in the diffusion of languages in a corpus containing two or more languages [33]. Hence it helps us to quantify the code switching in the dataset i.e. the number of Hinglish words in the dataset. A value closer to 1 indicates good code switching in the dataset. Let k be the total number of languages in the corpus and p_j is the fraction of the words in the language j in the corpus. Then MI is calculated mathematically by equation (2).

$$MI = \frac{1 - \sum_{j=1}^{k} p_j}{(k-1)\sum_{j=1}^{k} p_j^2} \tag{2}$$

Multilingual index was calculated reported as 0.601 for the HOT dataset and 0.684 for the IPM Dataset.

3.2.3 Fleiss Kappa

The Fleiss's Kappa [34] helps to determine annotation agreement between three or more raters. Since our dataset is triply annotated, hence we calculate the Fleiss's kappa metric on our dataset. Unlike Cohen's Kappa where all annotators need to annotate all the subjects, for Fleiss's Kappa all the raters need not annotate all the subjects. Let n be the number of subjects, k be the number of categories, m be the number of annotators for each subject and x_{ij} be the number of annotators that categorize subject i to category j. Then, Fleiss's Kappa is mathematically defined by equation 3.

$$k = \frac{p_a - p_e}{1 - p_e} \tag{3}$$

The Fleiss Kappa was calculated for inter rater agreement between the three annotators for the IPM dataset, and was reported as 0.782.

4. Methodology

Our methodology is composed of two parts. Classification of text and the classification of images. In this chapter we will discuss about each of the following in detail. Also, we discuss about deep learning techniques like long short term memory, CNN (convolutional neural networks) and basic supervised machine learning models like SVM and random forest classifier. Also, we elaborate upon the transfer learning aspect of deep learning.

4.1 Classification of Textual Data

We use an LSTM based model with transfer learning for this task. This deals with the problem of hate speech identification in a code-switched language pair of Hindi-English into namely three categories - Benign, Abusive and Hate Inducing. This paper establishes itself in the field of hate speech detection as state of the art work on HEOT dataset.

This involves the following steps: Pre-processing, Training of word embeddings, training of the classifier model and then using that on HEOT dataset. The complete pipeline is shown in the Figure 23.

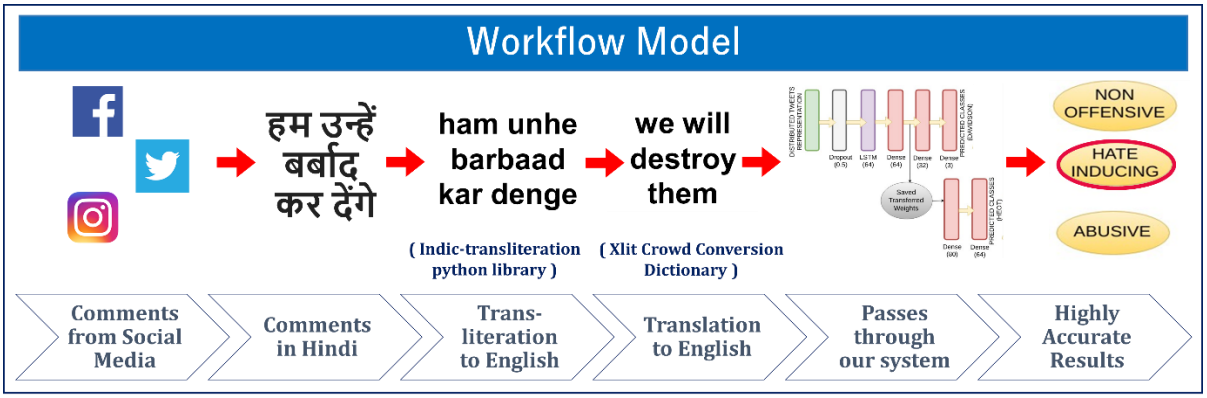


Figure 23 Workflow Model for text classification

4.1.1 Pre-Processing

In this project, two datasets were used. One of the dataset has been released by [31] which comprises of English language tweets and the other is HEOT dataset provided by [21].

The tweets obtained from datasets were channelled through a pre-processing pipeline with the final objective to transform them into feature vectors which involved the following steps :

(i) Removal of punctuations, mentions, emoticons, URLs, stopwords, numbers, hashtags, etc.

(ii) This was followed by translation of each word to English using Hindi to English dictionary i.e. Xlit-Crowd conversion dictionary.

(iii) Since Hinglish deals with a lot of spelling variations, the authors manually included some common variations of regularly used Hinglish words. The final dictionary comprised of 7200 word pairs.

(iv) Also, to deal with profane words, which are not present in Xlit-Crowd conversion dictionary, a profanity dictionary with 209 profane words was also created. An example is shown in Table 8.

Table 8 Examples of word-pairs in Hinglish-English dictionary

Hinglish	English	Hinglish	English
Acha	good	Gunda	Thug
S**la	Blo*dy	Ra*di	H**ker

4.1.2 Training Word Embeddings:

After the tweets were pre-processed, different word embeddings were trained on the datasets. Firstly, the GloVe embeddings [35] were used. Secondly, the Twitter word2vec embeddings [36], which is a 400 million tweets, word embedding model.

Both these were then trained on the datasets, namely HEOT and the one provided by [31]. These embeddings help to learn the distributed representations of tweets by creating word vectors. These word embeddings form the classifier model which then classifies the tweets in one of the three categories.

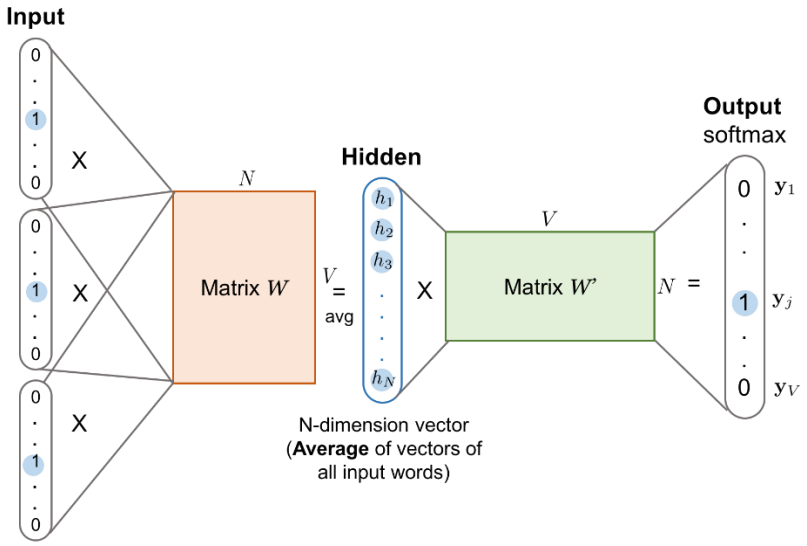


Figure 24 Concept of Word Embeddings

4.1.3 LSTM Model with Transfer learning

After the creation of word embeddings, the final training and testing of the dataset was done on an LSTM based ternary classification model which finally categorizes the tweets in one of the three categories – Benign (Non Offensive), Abusive and Hate Inducing. The model is depicted in Figure 25.

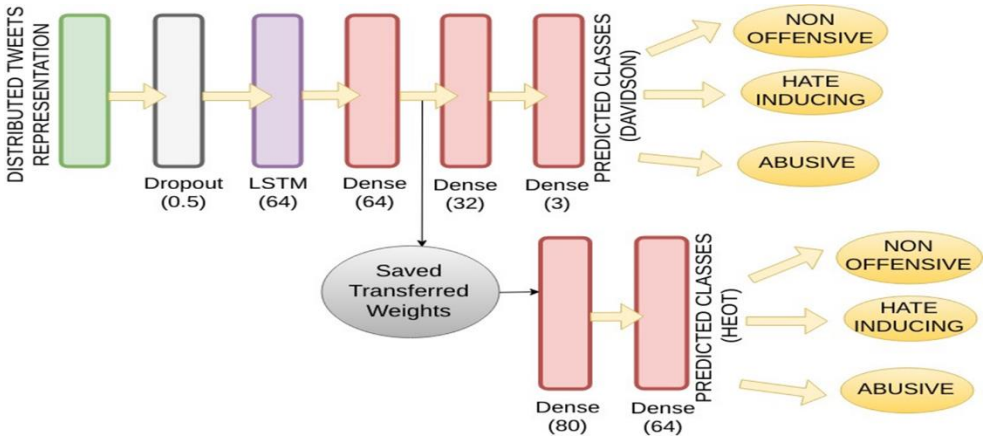


Figure 25 Architecture of LSTM model with Transfer Learning

The model takes the distributed tweets representation as the input which is the word embeddings. Next, a dropout layer is added to prevent overfitting of data. This has a value of 0.5. The model uses an LSTM layer and three dense layers. The LSTM layer has a dropout of 0.2 and the dense layer is of size 64, 32, 3 respectively. We have used Categorical crossentropy as the loss function in the last layer due to the existence of several classes. Also, to prevent overfitting, Adam optimizer along with L2 regularization has been used. To get the best set hyper-parameters and for the achievement of optimum results, extensive grid search was also done.

As we can note in Figure 25, the LSTM model was trained on the dataset provided by [31]. After passing through the first dense layer, the trained model is saved and with the help of transfer learning, the model is retrained on HEOT dataset. The use of transfer learning helps to improve the score by making use of the saved weights from the model trained previously on Davidson tweets dataset. The retrained model is then passed through a series of dense layers to finally distinguish the tweet in the one of the three categories.

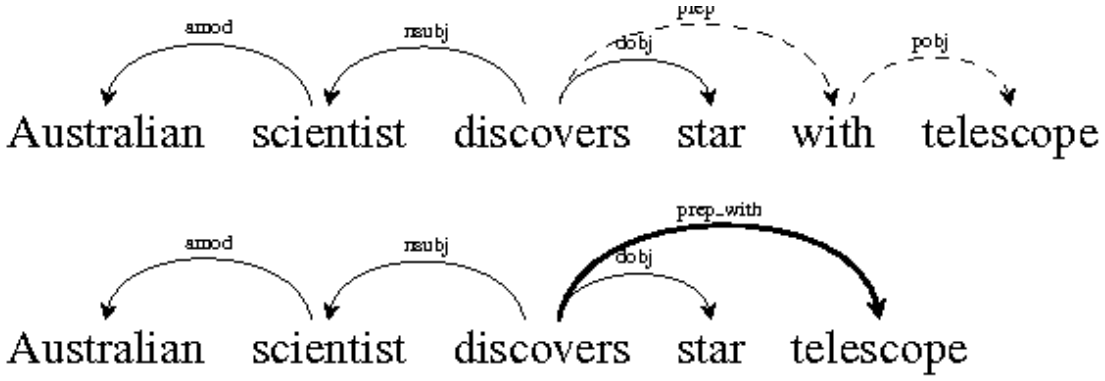
4.2 Dependency based parsing

The normal word2vec embeddings also called as vanilla based word2vec embeddings consider adjacent words in a sentence as a part of the context. Based on the window size which generally is kept fixed at 2, 3, 5, the context window tells the number of words in the window which will be considered while creating the word vector.

With the use of dependency based word embeddings we consider the context of the pair of words rather than adjacent words. Firstly, a parse tree is generated though the help of a dependency parser which is parsed on a context of a Hindi corpus of size of about 10GB. Once, the parse tree is generated, this parse tree consists of the various word pairs and their context with each other. Parse trees are passed to word2vecf which is currently available for English language to create a unique set of embeddings called "*Context Based Embeddings*" which no more consider adjacent words but rather on the basis on context. For example – Noun – Adjective pair or Verb - Adverb pair.

Smart embeddings produce better results than Vanilla Embeddings and prove much useful than the ordinary embeddings and this is shown by the results in a later stage.

Figure 26 shows the contrast between ordinary embeddings and the new embeddings.



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Figure 26 Vanilla Embeddings v/s Context Embeddings

4.3 Classification of Imageries

Our methodology primarily consists of the following steps: Preprocessing of dataset followed by training the classifier model and then using it on IPM dataset.

4.3.1 Preprocessing the dataset

The preprocessing consists of preprocessing of the images, extracting text from images followed by preprocessing of textual data.

4.3.1.1 Preprocessing Images

The image needs to be pre-processed in order to extract text out of the meme efficiently. Though the OCR performs some inbuilt pre-processing on the image, we perform the following steps for processing the image ourselves:

(i) Rescaling:

Some of the images need to rescaled to a larger size to enlarge the text written in a smaller font in the meme in order to make the text recognizable to the OCR. We use the resize feature of the OpenCV module [37] in python to rescale the images.

(ii) Gaussian blurring:

The blurring effect is used to reduce noise from the image. Gaussian blurring is performed by convolving the image with gaussian kernel. We use the Gaussian blur feature of the OpenCV module [37] to perform gaussian blurring.

(iii) Deskewing:

Some of the memes have text written at some skewed angle. Deskewing helps to rotate the image such that the text written in the image is mostly horizontal.

(iv) Gaussian adaptive thresholding :

Thresholding [38] converts the text into black and white format so that it is easily recognizable to OCR. In adaptive thresholding, the gaussian mean of the surrounding area determines the threshold value for the pixel of the image.

4.3.1.2 Extraction of text from images

After performing the above preprocessing on the images, we pass it through an open source OCR reader *ocr.space* (http://bit.ly/30uxnQ9) for extracting text out of the memes. Table 9 depicts the samples after text extraction from the image examples given above. Also, Table 9 shows the English translation of the Hinglish text extracted from the Memes.

Table 9 : Example of Hinglish text extraction from the memes depicted in Figure 21 with their respective English translations

Figure	Hinglish Text Extracted	English Translation	Label
Figure 1	Udi baba, Mark, homse kab milega ?	Hey Mark, when will you meet us ?	Non Offensive
Figure 2	Kisne kaha ki main pogo dekhta hun. Mummy se shikayat karunga	Who said that I watch pogo. I will complain to mother	Satirical
Figure 3	Ye to acha hai India mein beauty contest mein reservation nhi hoti.	It is good that there in no reservation in beauty contests in India (Derogatory remark on personal appearance)	Hate Inducing

4.3.1.3 Preprocessing Text from Images

The tweets obtained from data sources were sent through a pipeline with the objective to convert them into semantic feature vectors.

(i) Initially, the hashtags (For example: #indianpolitics), URLs, user mentions (denoted by '@') and numbers were removed from the text since they do not convey any relevant information about the sentiments of the text. Also, using NLTK library, the stop words were eliminated.

(ii) The emoticons (For example: ":)", "XD") were replaced by their textual description about the true emotions they depict.

(iii) Many of the comments which are in Devnagari (Hindi) script were converted to Roman (English) script. This was done using a python library called Indic-Transliterate (http://bit.ly/2JtSc95)

(iv) The Hinglish text now obtained is converted to their respective English translation using an Xlit-Crowd Conversion Dictionary.

(v) This is followed by the use of various word embedding representations such as FastText [39], Twitter word2vec [36], Glove [35] and Bert [40] embeddings for building the first layer of the LSTM side of the model which is the word-embedding layer. Different embeddings models are used to obtain the word vector representations of the preprocessed tweets. The embedding models are used one by one to figure out the best set of word embeddings.

4.3.2 Data Augmentation

Since the task of hate speech detection from images using deep learning requires large number of images, the technique of data augmentation is used to increase the size of the dataset to train the classifier model. The definition and the initial work for data augmentation was provided by [41]. We have used mainly five different types of augmentation techniques for our work which has significantly helped our model to train better on the dataset and provide much better results than what were observed previously.

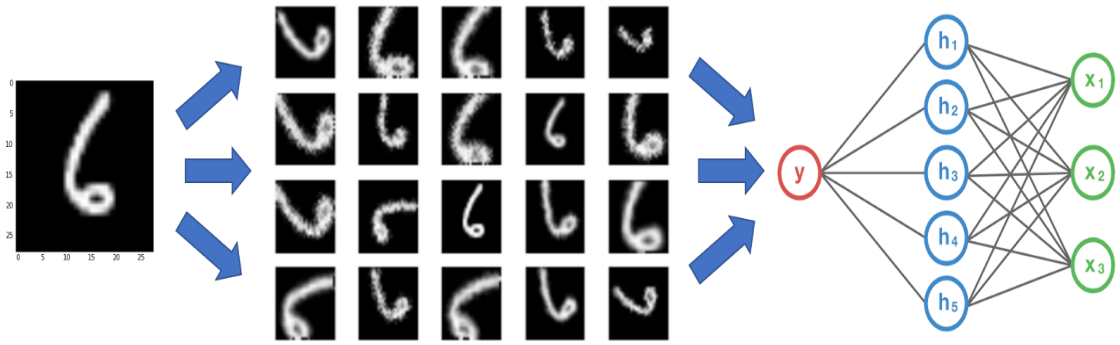


Figure 27 Data Augmentation Technique

The data augmentation techniques used are :

(i) Scaling : We scale the image both inwards and outwards for creating new images in the dataset.

(ii) Translation : The images are moved in the X or Y direction by varying degrees.

(iii) Rotation : Rotation is performed at 90 degrees, 180 degrees and 270 degrees.

(iv) Flipping : The images are flipped in both the horizontal and vertical direction.

(v) Adding noise : Gaussian noise is added to distort the high frequency features that are not useful for the model.

For implementing the above techniques, we have used the various functions from the *ImageDataGenerator* class of the Keras image processing python library.

4.3.3 Proposed CNN-LSTM Model

We propose a model which is a binary channeled CNN cum LSTM model which takes text in the form of word vector representation and image as its input and finally concatenates the two channels to produce the final result. The model architecture is depicted in Figure 28.

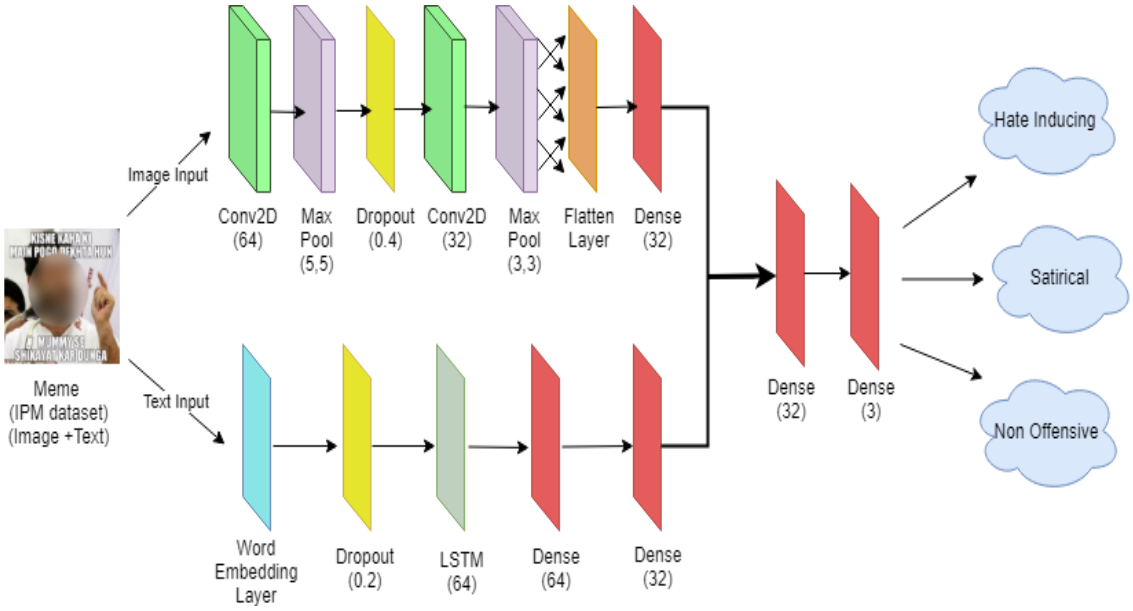


Figure 28 Architecture of Proposed CNN-LSTM model

The CNN channel

The CNN channel processes the image and tries to extract certain feature of the image that would help to classify the meme into one of the three categories i.e. hate inducing, satirical and non offensive. The pre-processed images form the input to the first layer of the CNN channel which is a convolution2D layer with filter size 64, kernel size as (5, 5) and activation function of Relu [42]. This is followed by a max_pooling layer of size (5, 5). The convolution layer helps to create a convolution kernel that is convolved with the input layer to produce a tensor of outputs. Next, we employ another convolution2D layer, this time of size 32, kernel size (3, 3) and activation function as Relu, and a max_pooling layer of pool size (3, 3). This is succeeded by a flatten layer that converts the 3 dimensional feature map to 1 dimension. We also use a dropout layer of size 0.4 to prevent overfitting of data. This is followed by a dense layer of size 32. The CNN channel tries to utilize the image features to decide in which category the meme is to be classified.

The LSTM channel

This particular channel is for the textual content that has been extracted from the image and preprocessed during the earlier stages of the work. The first layer of the LSTM channel is the embedding layer which takes word vector representation of the extracted caption from the Meme image. These embeddings help to learn distributed representations of captions. After experimentation, we kept the size of embeddings fixed to 100. Different embedding models unravel the different aspects of the language. For example, the dependency parser focuses on the similarity between the two terms. On the other hand, statistics of bag of word (BoW) embeddings emphasize on the word associations. Some of the embeddings used are FastText [39], Glove [35], Twitter word2vec [36] and Bert [40]. The embedding layer is followed by a dropout layer of size 0.2 to prevent overfitting of data. The next is the LSTM layer of size 64 with dropout of 0.4. The LSTM layer is followed by two dense layers of sizes 64 and 32. This part of the model serves as a processing model for the textual content.

Recombination of channels

The two parallel channels of the model which process the text and images separately are finally recombined to a single channel to obtain the final results. The concatenation is followed by the presence of two dense layers of sizes 32 and 3. The last dense layer of size 3 uses Softmax as the activation function. We use L2 regularization and Adam optimizer [43] for preventing overfitting. The loss function used in the last layer was categorical cross-entropy which serves beneficial in the case of multiple classes. The output obtained after passing through the dense layers is one of the three classes, i.e., Hate Inducing, Satirical and Non offensive.
The two parts of the model therefore help to process the text and image in parallel thereby tackling the task of hate speech detection in a much formal and organized manner.

4.4 Supervised machine Learning models for Classification

An ensemble consists of a set of individually trained classifiers (such as Support Vector Machine and Classification Tree) whose predictions are combined by an algorithm. Ensemble methods is expected to improve the predictive performance of classifier.

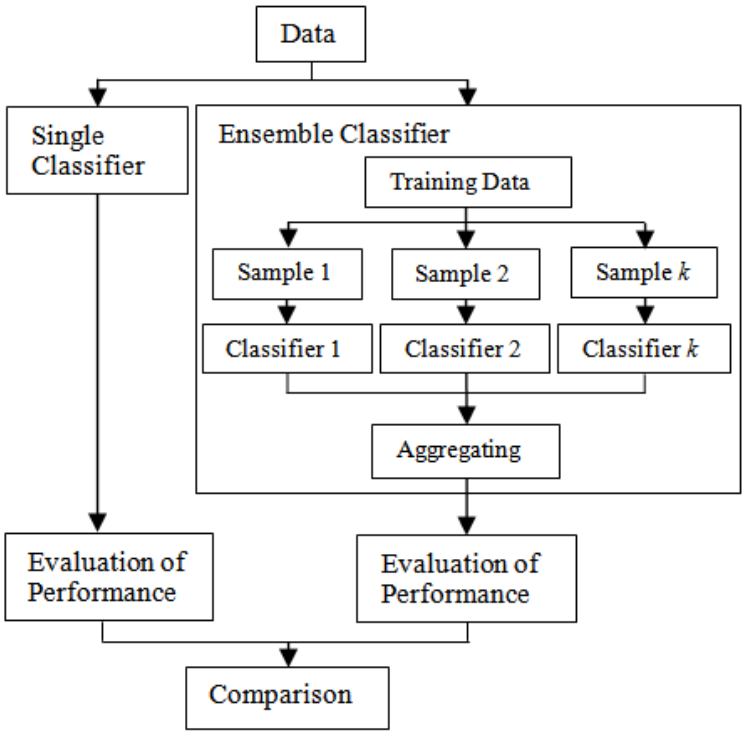


Figure 29 The workflow of research methods

4.4.1 Support Vector Machines (SVM)

A Support Vector Machine (SVM) may be a discriminative classifier formally outlined by a separating hyper plane. In different words, given labeled training information (correct labels), the classifier outputs the best hyper plane found that categorizes new examples into the given labels. In a two dimensional space this hyper plane can be visualized as a line bisecting the plane into two parts where in each class lay in either facet.

The learning of the hyper plane in linear SVM is completed by reworking the problem using some linear algebra. This is where the role of kernel comes in. Given a linear kernel, equation 4 defines how the prediction for a new input is calculated by doing a dot product between the input (x) and each support vector (x_i).

$$f(x) = B(0) + sum(a_i * (x, x_i)) \quad (4)$$

The coefficients B_0 and a_i, for every input (x) are estimated by the learning algorithm from the training information.

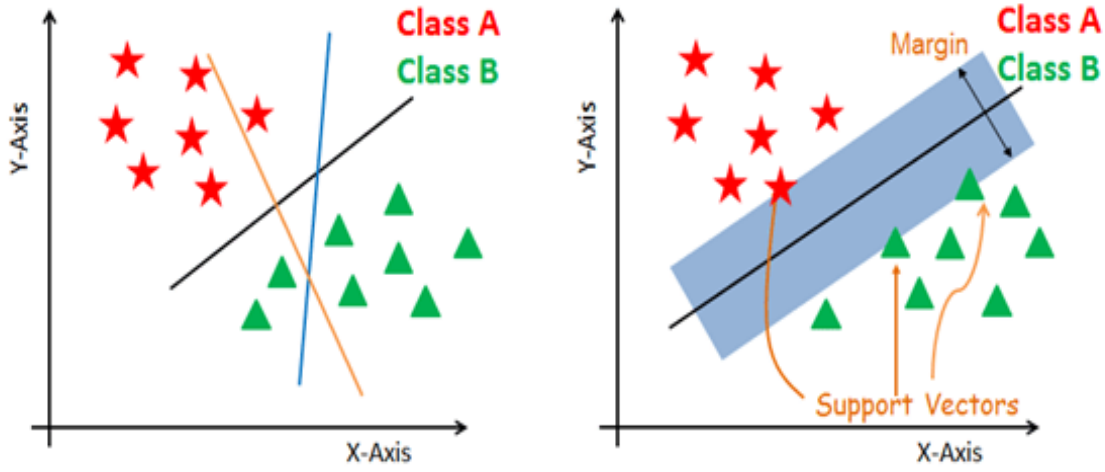


Figure 30 Support Vector Machines (SVM)

4.4.2 Random Forest Classifier

An ensemble method for classification, regression and other tasks is performed using a Random Forest (RF) classifier. Random forest operates by creating a swarm of decision trees at the training time and categorizing the label into a class which is the mode of the classes of the separate trees. Random Forest classifier architecture is shown in Figure 31.

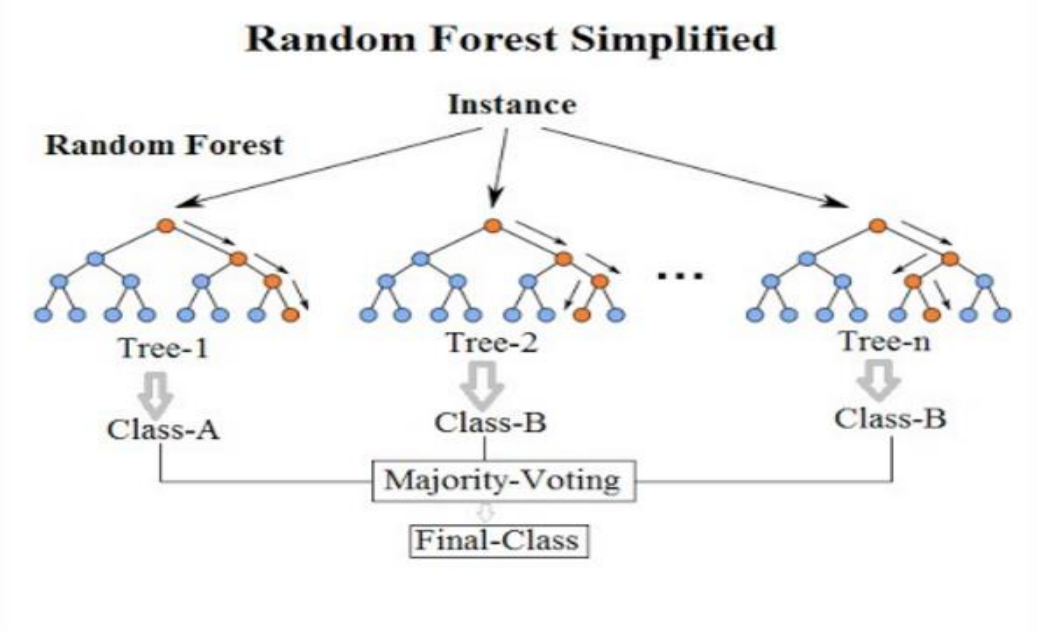


Figure 31 Random Forest Classifier

4.4.3 Gradient Boosting

Gradient boosting is an algorithm that can be used effectively for classification and regression problems. Gradient Boosting also develops a prediction model in the form of decision trees, which are a collection of weak prediction models. These predictive models are built in a stage wise fashion similar to other boosting methods. Similar to all the boosting algorithms, gradient boosting algorithm also pools weak learners into a single strong learner iteratively. Gradient boosting in decision trees is shown pictorially using Figure 32.

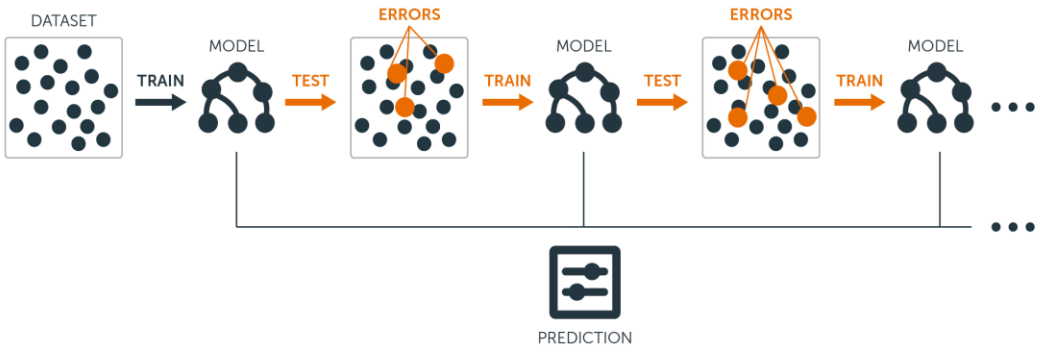


Figure 32 Gradient Boosting in Decision Trees

4.5 Deep Learning Techniques for Classification

Deep learning (hierarchical learning or deep structured learning) belongs to the family of methods relying on learning data representations and machine learning algorithms. Deep neural networks and recurrent networks constitute the deep learning architectures. It consists of nodes that are interconnected to each other and are similar to that of a human brain. The architecture is similar to the brain neurons. This architecture is described in the Figure 33. In this, the neuron is depicted by the node and the connection between the artificial neurons is shown with the help of arrows. In this subsection we study about:

- Long Short Term Memory (LSTM)
- Convolution Neural Network (CNN)
- CNN - LSTM
- Max Pooling
- Transfer Learning

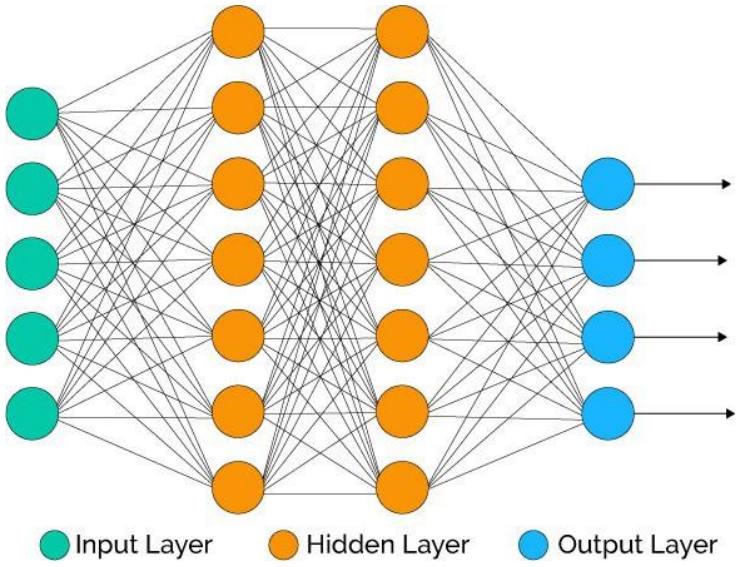


Figure 33 Neural Network Architecture

4.5.1 Long Short-Term Memory Networks (LSTM)

Cells of the long short-term memory (LSTM) model are actually constituted by recurrent neural nets (RNN). In one unit of LSTM, the components are:

- Functionality Cell
- Input Gate
- Forget Gate
- Output Gate

This is depicted in Figure 34. The functionality cell is performs the basic functionality and remembers the values for some time and hence constitutes the memory of the model. The gates are kind of artificial neurons like in any other deep learning model. The normally used sequential models are also called feed forward neural networks while LSTM based models are called feedback networks. Weighted sum of activation function like Relu, sigmoid is calculated by the gates and the over-all flow is regulated by the help of the gates. The input gate controls the input flow. The output gate controls the output flow. The function of forget gate is also similar to these gates.

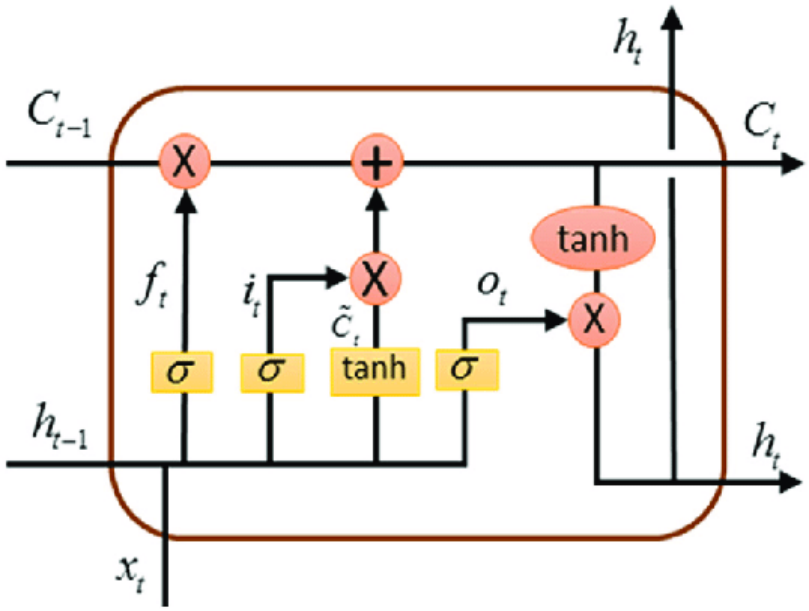


Figure 34 Long Short-Term Network

4.5.2 Convolution Neural Networks

The deep learning model that is the most suitable for image processing is the Convolution Neural Network or commonly called CNN. In this the model takes the image as an input in the form of a vector, and on the basis of weights and priority of various features, processes the image and finally distinguishes it. It is very similar to the human brain's neurons and are basically an imitation of it. Very minimal preprocessing techniques are

required in CNN because in other techniques the features need to be extracted which require further a lot of processing of the data which is not the case with the images. It can learn the image features mostly by itself.
The basic operation done is the convolution of different windows. The windows size is fixed in the beginning and then we do convolution operation among the two windows. It can be of 2 types:

- Convolution1D
- Convolution2D

And both of them have their own functionalities which make them distinct to each other.

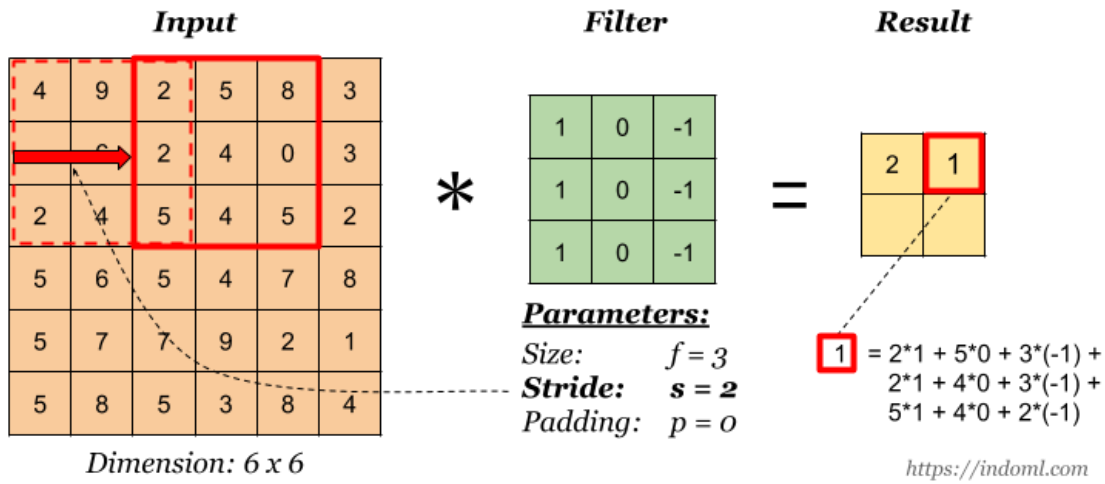


Figure 35 Working of Convolution Neural Network (CNN)

4.5.3 Max Pooling

Max pooling uses the basic technique of sampling. Our primary goal in max pooling is to firstly sample the image or the features generated by CNN or other layers. Once this is done, the dimension is reduced by the process of sampling on the windows or the subregions that are formed. The image can be represented directly or by the means of a matrix or features of a previous layer.

Various symbolic representations reduce the computing cost as the parameters to the functions are drastically lowered. To remove the problem of overfitting, vague form of the symbols and representations are used.

The window selected in the process of max pooling is selected as such there is no overlapping and the maximum of that window is chosen and through the help of this, the dimension is reduced and overfitting is eliminated.

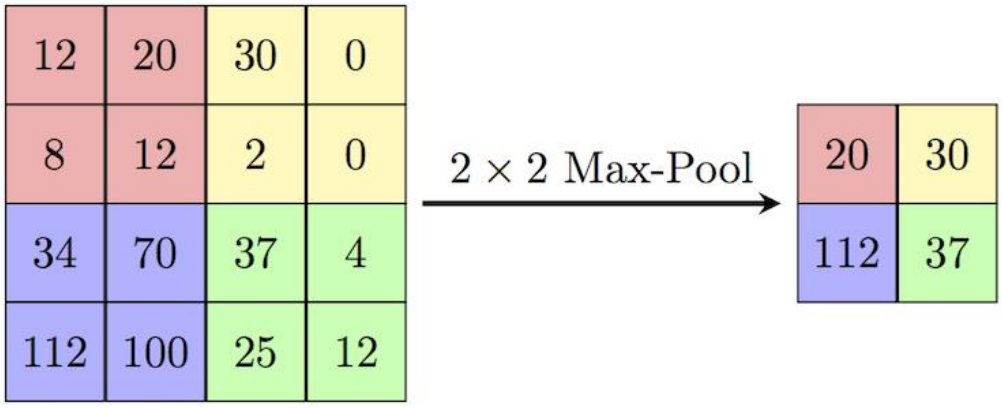


Figure 36 Max Pooling Layer

4.5.4 CNN-LSTM

The CNN and LSTM models are often combined with each other to create a single model like in our case of offensive memes classification. When there is a multimodal analysis (like: Images, video, audio and their combinations), or some kind of activity analysis, the CNN-cum-LSTM model proves its dominance over the others in a much profound manner.

In this the CNN channel proves helpful for the aspects such as image processing. The LSTM part of the model proves its mettle in cases of contextual data like text classification. Together the combination of CNN and LSTM produces optimum results for complex problem space.

4.5.5 Transfer Learning

The methodology, Transfer learning works in a manner much similar to its name. By the transferring of information from one set of problem to a different set of problem is the basis how transfer learning is performed. This is generally used in a complex domain space where the amount of data is very less as compared to what is required.

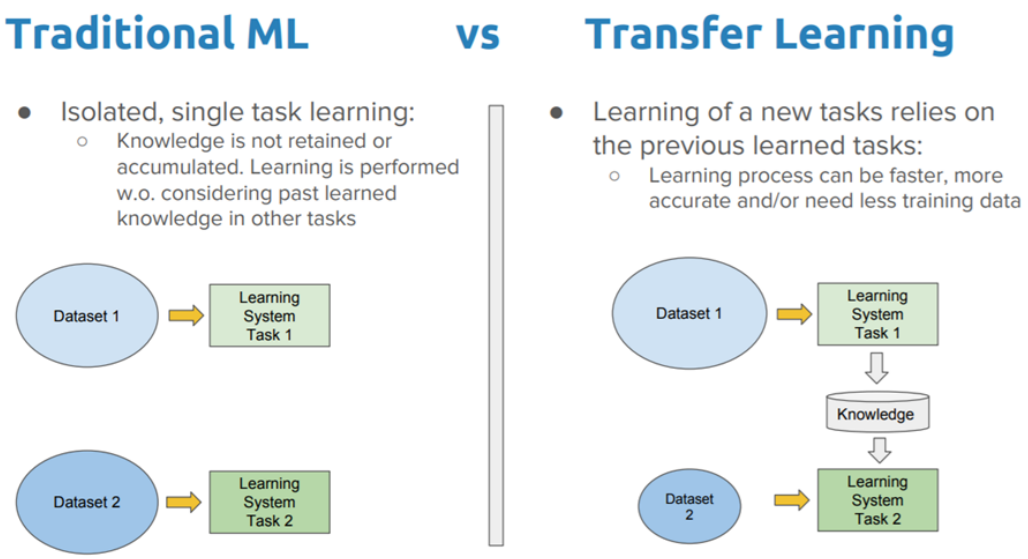


Figure 37 Contrast between Traditional ML and Transfer Learning

Starting problem which is also called the source problem deals with training of a model. Once the model is trained on a set of hyperparameters and a particular dataset, the weights so obtained by the process are saved and the same trained weights are then then used by changing the model hyperparameters and dataset or possibly some of the layers of the model after which the model is retrained on the dataset again to produce much better results than before. Finally, the target task which is the final task for which the model is created produces optimum results by the simple technique of transfer learning.

The NLP tasks such as sentiment analysis is one of the major applications where transfer learning is being used nowadays.

5. Experimentation

5.1 Tweets Classification

We compare our LSTM model with different word embeddings, namely Twitter Word2vec [36] (ii) Glove [35] (iii) Fastext [39] and (iv) Bert [40]. We discuss these word embeddings in detail in this section.

5.1.1 Word2vec: Skip gram and Bag of words

Word2vec is the technique or model to produce smart word representations using word embeddings. It can capture a great amount of accurate semantic and syntactic word relationships. Word2vec produces word embeddings using a shallow two-layered neural network. There are two methods for developing word2vec (both involving Neural Networks): Skip Gram and Common Bag of Words (CBOW). Both skip-gram and common bag of words convert unsupervised representation to supervised form for training the model. Figure 38 shows the example of skip gram and CBOW word2vec embeddings.

CBOW Model: CBOW model inputs the context of each word and tries to predict the word corresponding to the surrounding words in the window.

Skip Gram: Skip-Gram is the opposite of CBOW, which implies that given the word it predicts the context from the word.

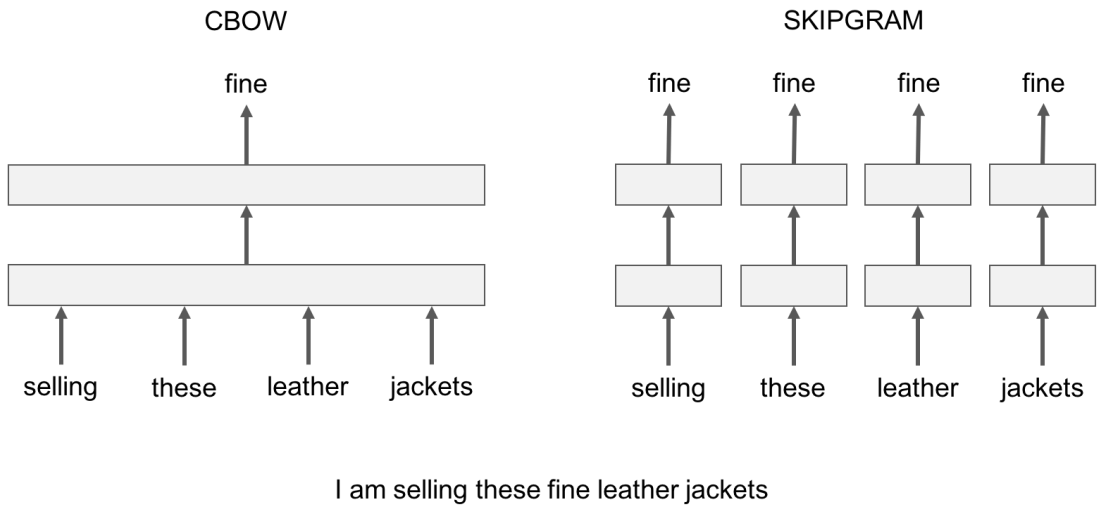


Figure 38 Example of Skipgram and CBOW Word2vec embeddings

5.1.2 GloVe embeddings

GloVe is an unsupervised machine learning algorithmic program for getting vector representations for words. Training of the embeddings is done on an aggregated global word to word co-occurrence statistics from a corpus, and the resulting word representations show great linear substructures of the vector space of the word. The similarity metrics used for nearest neighbor evaluations manufacture one scalar that quantifies the connectedness of two words. This simplicity can be challenging since two given words always exhibit a lot greater tricky relationships than can be captured by a single word. For example, *man* may be regarded as similar to *woman* as both words describe human beings; but, the two words can also be considered opposites because this is a primary reason of how human are different from each other.

In order to detect in a quantitative way the nuance important to differentiate man from *woman*, it is necessary for a model to associate more than a single number to the word pair. Figure 39 shows the example of glove word embeddings.

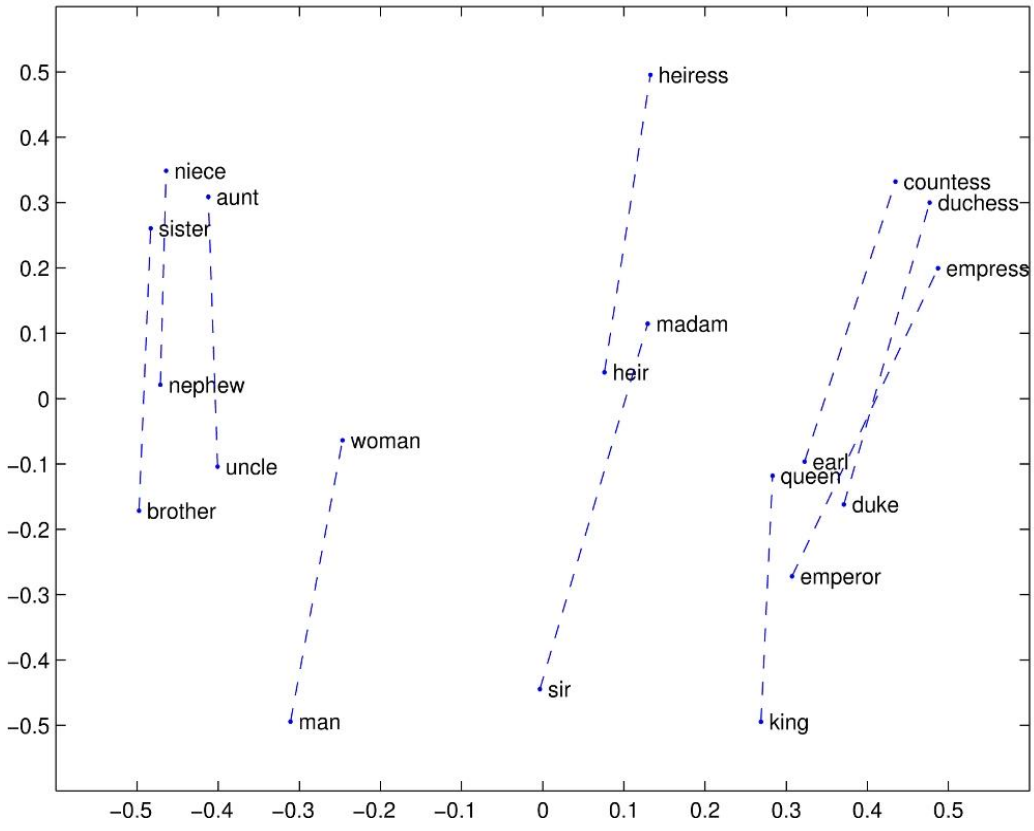


Figure 39 GloVe embeddings

5.2 Memes Classification

In this section, we analyze the experimental settings for analyzing the results on IPM dataset. Firstly, as baseline we conduct experiments using supervised machine learning models, namely Support Vector Machine and Random Forest classifier which will define the baseline results. We then try LSTM and CNN based deep learning models on the IPM dataset, followed by the analysis on our proposed model.

5.2.1 Baseline

The baseline model was created using a Support Vector Machine (SVM) and a Random Forest (RF) classifier. These two classifier models were trained using k-fold cross-validation with 10 splits. For the SVM classifier we choose kernel value as `poly' with the default value of degree = 3 as hyper parameters. All other hyper parameters for the SVM classifier are used at their default values. For the RF classifier after fine tuning the model, the hyper parameters chosen were n_estimators as 600, max_depth as 12 and max_features as *log2*. Figure 40 shows the system architecture for using the baseline models on the IPM dataset. Figure 41 shows the hyper parameter tuning graphs for random forest classifier.

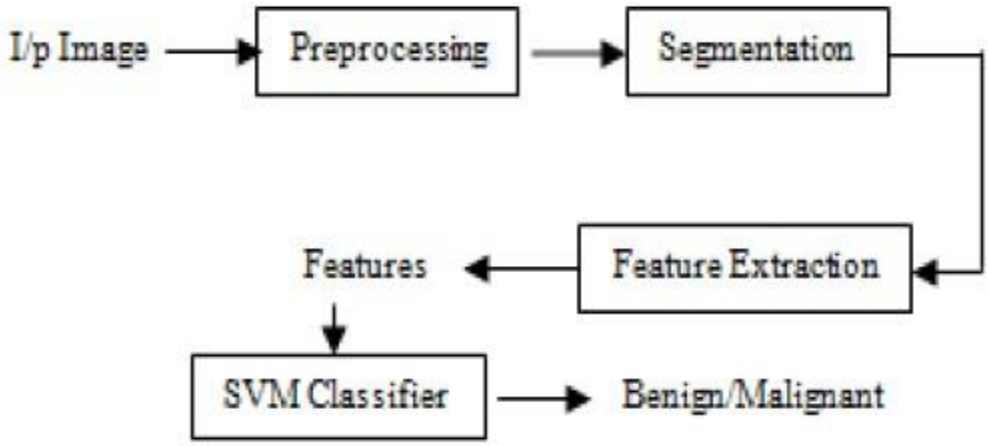


Figure 40 System architecture for baseline models

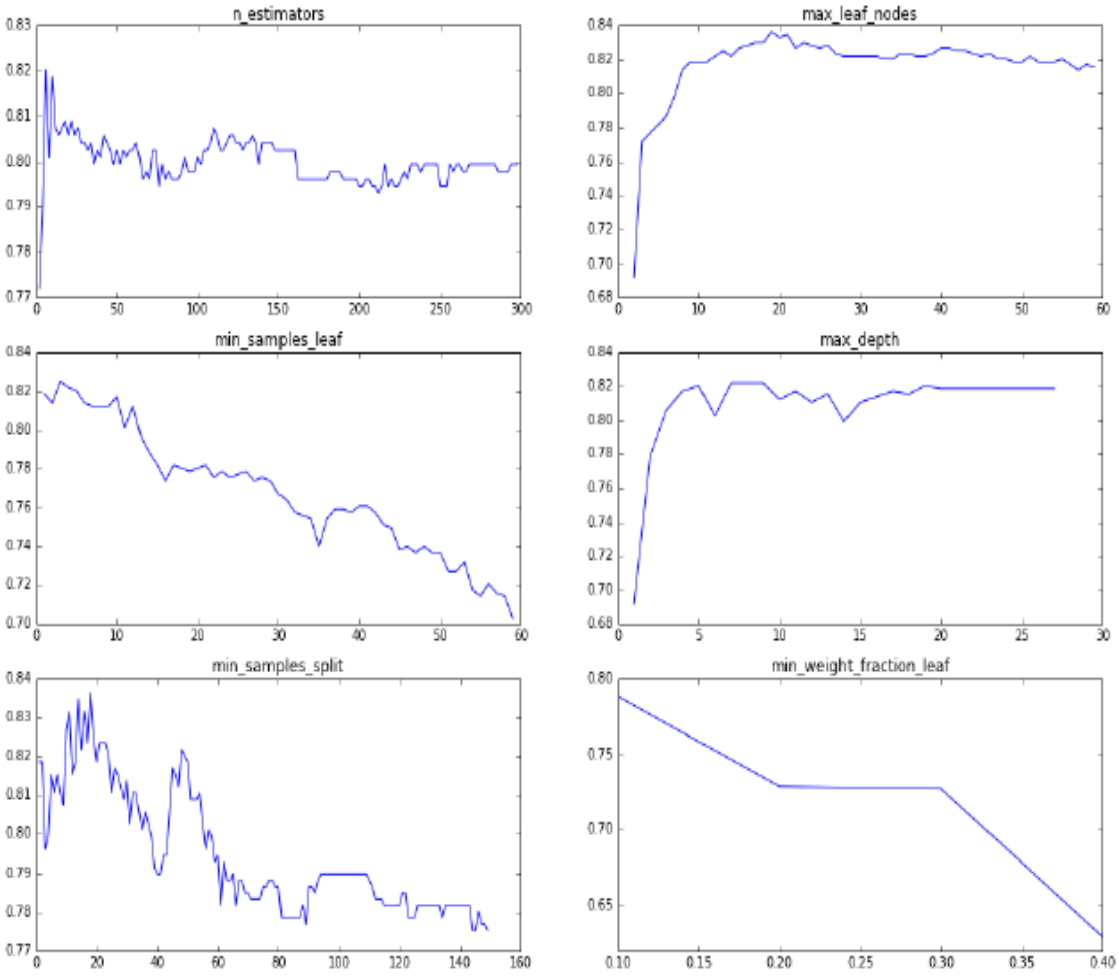


Figure 41 Hyper parameter tuning for Random Forest Classifier

We choose the following features from the images to be input in the baseline classifying models:

(i) GLCM feature: Gray-Level Co-occurrence Matrix [45] helps to determine the texture of the image which is useful for determining the emotional expression in an image. The GLCM functions characterize the feel of a picture by scheming how frequently pairs of pixel with specific values and during a fixed spatial relationship occur in a picture, producing a GLCM matrix, and so extracting applied mathematics measures from this matrix.

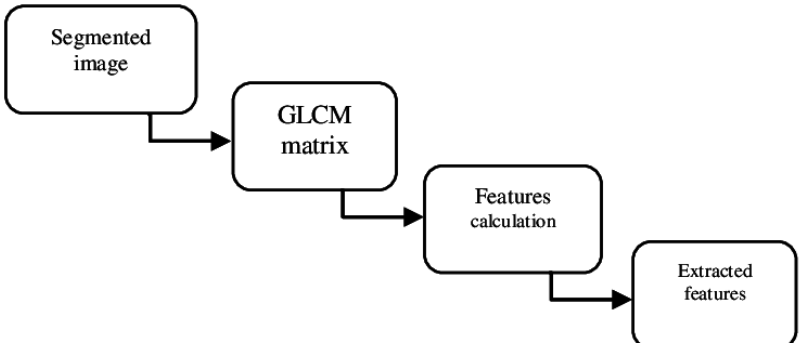


Figure 42 Using GLCM features

The texture of an image can easily be found with the help of these statistics. We use GLCM for determining:

(a) *Contrast:* Local dissimilarities are measured using the gray-level co-occurrence matrix.

(b) *Correlation:* Joint probability incidence of the given pixel pairs is measured.

(c) *Energy:* Sum of squared elements in the GLCM is provided using the energy feature.

(d) *Homogeneity:* Closeness of the scattering of elements in the GLCM to the GLCM diagonal is calculated.

These features are used as the input to the classifier models that uses the texture of the image to classify the meme into any of three categories.

(ii) Colorfulness feature: Color is one of the important ways to convey a message through a image. However, mapping low-level color options to emotions may be a complicated task that should take into account theories about the employment of colors, psychological feature models and involve cultural and social science backgrounds. In alternative words, users from completely different cultures or backgrounds would possibly understand and interpret identical color pattern quite contrarily. We use Earth Mover's Distance (EMD) between the histogram of an image and the histogram having a uniform color distribution [46] for calculating the colorfulness in an image. We have extracted a number of unique colors as a feature for classifying memes in the baseline model of SVM and Random Forest (RF). Since colors are used politically to spread religious hatred among people, so color can be an important feature for the classifying model.

(iii) Tamura features: Tamura features also help for determining texture of an image as shown by [47]. The emotion of a image can easily be calculated using the texture feature of the image. Professional photographers and artists typically produce footage that is sharp, or wherever the major object is sharp with a blurred background. But we tend to ascertained that additionally blur in footage are often used with efficiency to attain a desired expression. We use coarseness and directionality as Tamura features for input to the classifying model.

(iv) **Human face feature:** Human faces are important for drawing attention to a meme. We detect frontal faces (if there are any) in the picture. This feature is used because two similar images can have different impact on the user depending on the feature that it contains a face or not. The example is shown in Figure 43. The number of human faces and the size of human faces are used as features to our classifying model. [48].

Figure 43 Two similar images of a creature swimming in the sea, but different emotional impact due to the presence of human face.

We use these features as input to the classifying SVM and RF model with hyper parameters mentioned above.

5.2.2 Deep Learning Models

We also compare our model to some of the deep learning models. We use two deep learning models for validating the accuracy of our own model. The first model we experimented with is a CNN based model which is popular for image classification. The

CNN model was trained using a k-fold cross validation with 10 splits. The first layer of the CNN model is a Convolution2D layer with filter size 64 and kernel size = (5, 5). The activation function used is *Relu* [42] and a max_pooling layer of pool size = (5, 5) is employed. This is followed by another Convolution2D layer with filter size of 32 and kernel size of (3, 3), succeeded by a max_pooling layer of pool size = (3, 3). We also employ a dense layer of size 64 and a dropout layer of 0.4 to prevent over fitting. The hyper parameters are chosen with the help of grid search which helps to select those hyper parameters that produce the most optimal results.

The second model experimented is an LSTM based model which consists of an embedding layer, followed by a LSTM layer and two dense layers of size 64 and 32. Adam optimizer and L2 regularization are used to prevent overfitting of data. The dropout layer of size 0.4 is also added to prevent overfitting of data. The LSTM based model is also trained using k-fold cross validation with 10 splits. The loss function used for both the CNN based model and the LSTM based model is categorical crossentropy.

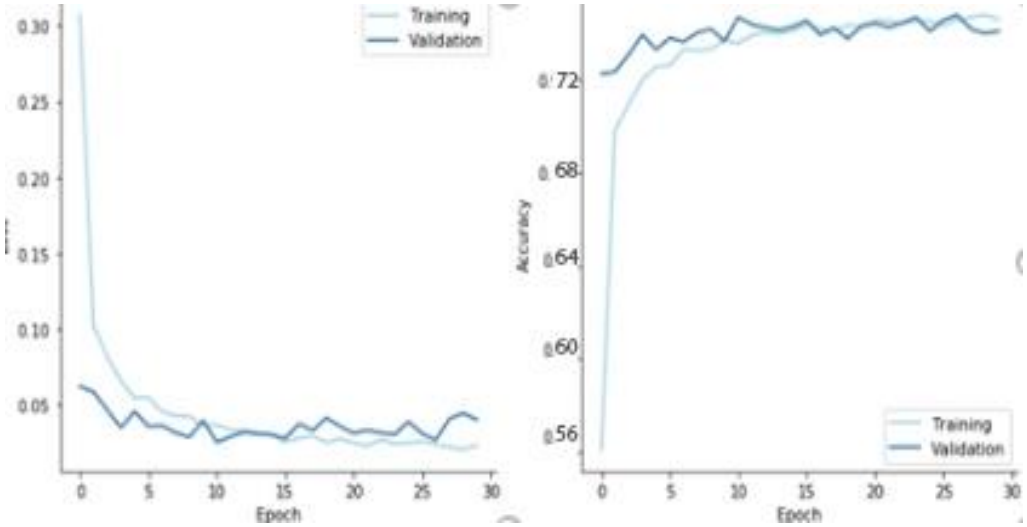


Figure 44 Hyper parameter tuning for LSTM

5.2.3 Proposed model

Now we compare the results of the baseline (SVM and RF) model and the deep learning models (CNN and LSTM) to our proposed model. Our model tries to incorporate the best features of both the CNN and LSTM deep learning models by considering images as well as the extracted text from the images for the classification task. We propose a binary

channel model in which the LSTM channel processes the text written inside the images and the CNN channel processes the image itself. The two channels are combined to produce the final results. We conduct the experiments on our model using different flavors of word embeddings i.e. (i) Twitter Word2vec [36] (ii) Glove [35] (iii) Fastext [39] and (iv) Bert [40] embeddings as well as the combination of the above embeddings. Many different sizes of embeddings were tested.

Finally, the size of the embeddings was chosen to be 100. Our model is also trained using k-fold cross-validation with 10 splits to maintain consistency in all the experiments.

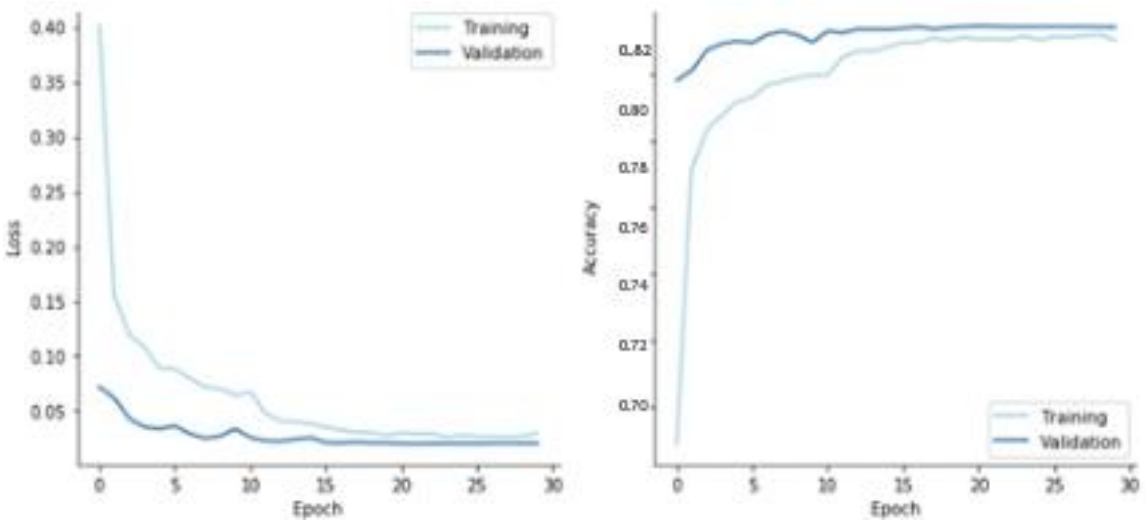


Figure 45 Hyper parameter tuning for proposed CNN-LSTM model

6. Results

6.1 Evaluation metrics used for calculation of results

The Baselines and Deep learning models described above are compared with each other in terms of the following metrics:

$$Accuracy = \frac{t_p + t_n}{t_p + t_n + f_p + f_n} \tag{5}$$

$$Precision = \frac{t_p}{t_p + f_p} \tag{6}$$

$$Recall = \frac{t_p}{t_p + f_n} \tag{7}$$

$$F1 - Score = \frac{2t_p}{2t_p + f_p + f_n} \tag{8}$$

where,

t_p = count of true positives

t_n = count of true negatives

f_p = count of false positives

f_n = count of false negatives.

6.2 Text Classification Results

We use an LSTM model with word embeddings trained on Glove [35]. Transfer learning was also used to benefit from the transferred weights that were trained on the previous dataset. Table 10 and Table 11 shows the performance of LSTM model after getting

trained on [31] with two types of embeddings in comparison to the models by [21] and [31] on the HEOT dataset. The results were averaged over three runs. We also compared the results on pre-trained word2vec embeddings. As depicted in the table, the custom trained Glove embeddings performs better than all other word embeddings. The results were also compared for the Davidson model which produces great accuracy for English language tweets. The model by [31] gives an accuracy of 90% of Davidson English tweets dataset while giving an accuracy of 57%, which indicates that model finds it hard to deal with the intricacies of Hinglish semantics. The LSTM model with transfer learning on the hand gives a comparable accuracy of 89% on Davidson dataset and also performs well on the HEOT dataset [21] resulting in an accuracy of 87% which is the state of the art results on HEOT dataset.

Table 10 Comparison of accuracy scores on HEOT dataset

Model	**Accuracy**
Davidson et al.	0.57
Our Model with embeddings trained on Glove	**0.87**
Our Model with embeddings trained on Word2Vec	0.82
Our Model with pre-trained Word2Vec embeddings	0.59
Mathur et al.	0.83

Table 11 Comparison of accuracy scores on Davidson Dataset

Model	**Accuracy**
Davidson et al.	0.90
Our Model with embeddings trained on Glove	**0.89**
Our Model with embeddings trained on Word2Vec	0.86
Mathur et al.	0.75

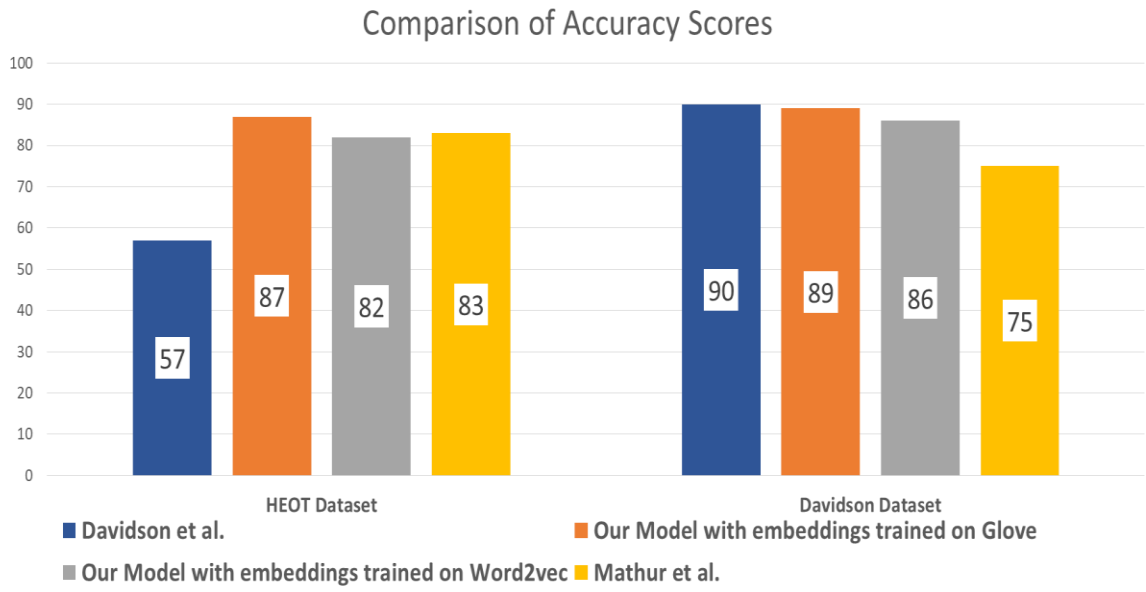


Figure 46 Comparison of accuracy score on HEOT and Davidson Dataset

6.3 Dependency based Embedding Results

The results of the new set of embeddings thus created with the help of dependency parser is shown in Table 12. The results are being shown for word2vec SG and word2vec CBOW embeddings of context window size as 2 and embedding dimension as 100. These is about 6% improvement in the case of SG dependency embeddings while the CBOW set of embeddinngs show a huge rise of about 7%, which is very significant as we can see from the results.

Table 12 Results using dependency based word embeddings

Embeddings				Accuracy	Precision
Embeddings	**Window Size**	**Dimension**	**Embeddings**		
SG	2	100	Vanilla	82.56 %	0.81
SG	2	100	Dependency	88.14 %	0.89
CBOW	2	100	Vanilla	79.68 %	0.80
CBOW	2	100	Dependency	86.01 %	0.88

6.4 Image Classification Results

6.4.1 Results via SVM and Random Forest classifier

We use these features as input to the classifying SVM and RF model with hyper parameters mentioned above. The results using these baseline models and the features mentioned are shown in Table 13. We use precision, recall and F1-score as the metrics for determining the baseline results. It was seen that SVM performs marginally better that than the Random forest classifier when using GLCM features. The SVM classifier also gave comparable results when using the human face features as the input. This forms our baseline results for Hinglish offensive memes classification on IPM Dataset.

Table 13 Baseline results for non-offensive, hate-inducing, satirical memes classification on IPM dataset using SVM and random forest classifier with different features

Feature	GLCM		Colorfulness		Tamura		Human Face	
Classifier	SVM	RF	SVM	RF	SVM	RF	SVM	RF
Precision	**0.622**	0.584	0.542	0.475	0.562	0.512	0.608	0.545
Recall	**0.651**	0.597	0.522	0.457	0.592	0.538	0.619	0.595
F1 - Score	0.634	0.575	0.573	0.514	0.583	0.546	**0.658**	0.603

6.4.2 Results of Deep learning models

We compare the results of the two deep learning models in Table 14 using precision, recall and F1-score as the metrics. The CNN based model produces much better results than the LSTM based model, while the results are marginally better than the SVM baseline model described above. This is due to the fact that hate speech can be conveyed in the form of text as well as images. Analysing the image solely, therefore does not produce great results for this classification task. Hence, we see that the features extracted manually and fed to SVM or RF classifier gives comparable results to that of CNN or LSTM based models.

As proposed in our model, we analyse both image as well as the text extracted to produce the final results.

Table 14 Results of Deep learning models on IPM dataset

Result	CNN Model	LSTM Model
Precision	0.632	0.581
Recall	**0.674**	0.534
F1-Score	0.618	0.604

6.4.3 Results of binary channel CNN-LSTM model

The results of our model using different types of word embeddings on the IPM dataset are shown in Table 15. The results are compiled using precision, recall and F1-score as metrics of evaluation. Our model outperforms the baseline (SVM and RF models) and also the deep learning (CNN and LSTM) models, hence establishing itself as the state of the art for the task of offensive memes classification in Hinglish language. As seen from Table 15, best results obtained using a single embedding model was with the Glove embeddings. Recall score of 0.822 was recorded with Glove embeddings. Also experiments were conducted using the combination of word embeddings where (Glove + Fastext) is seen to produce the best results. Here, precision of 0.803 was obtained which demonstrates that model outperforms all the other models on IPM dataset for the task of offensive memes classification in code switched language (Hinglish).

Table 15 Results of our model with different flavors of word embeddings.

Features	Precision	Recall	F1-Score
Glove(Gl)	**0.812**	**0.816**	**0.822**
Twitter Word2Vec(Tw)	0.781	0.786	0.781
FastText(Ft)	0.791	0.784	0.793
Bert(Bt)	0.758	0.804	0.784
(Gl) + (Tw)	0.727	0.751	0.722
(Gl) + (Ft)	**0.803**	**0.811**	**0.810**
(Bt) + (Tw)	0.788	0.780	0.772
(Gl) + (Ft)	0.779	0.790	0.765
(Bt) + (Ft)	0.780	0.783	0.796

6.5 Demo Screenshots

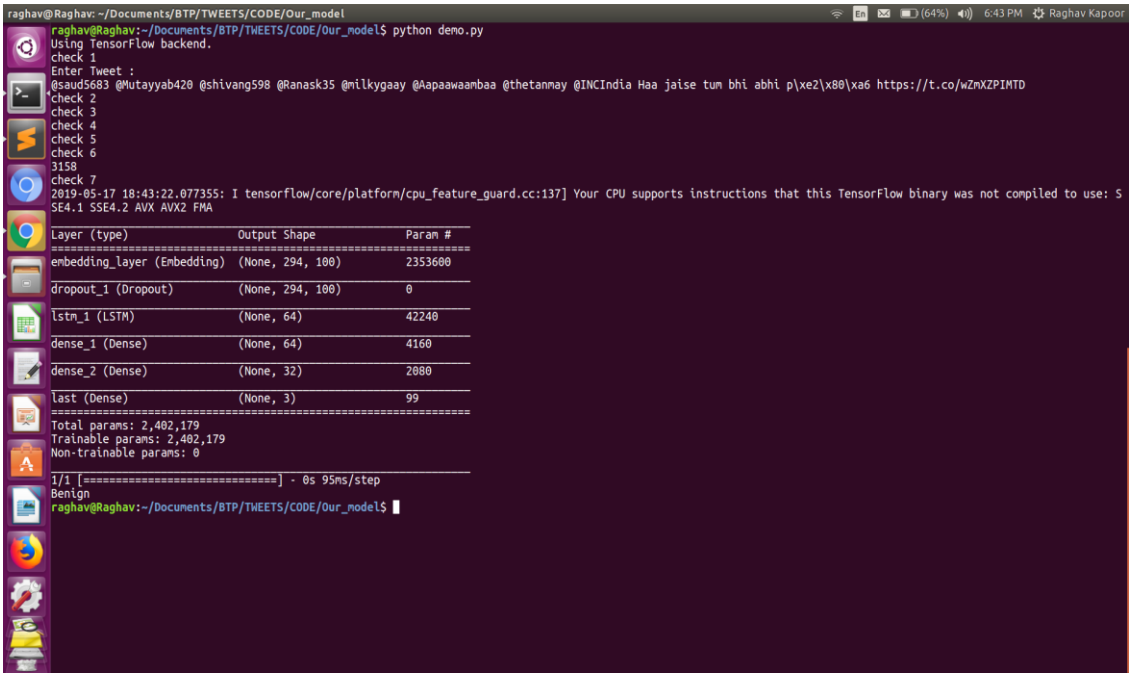


Figure 47 Classification Example

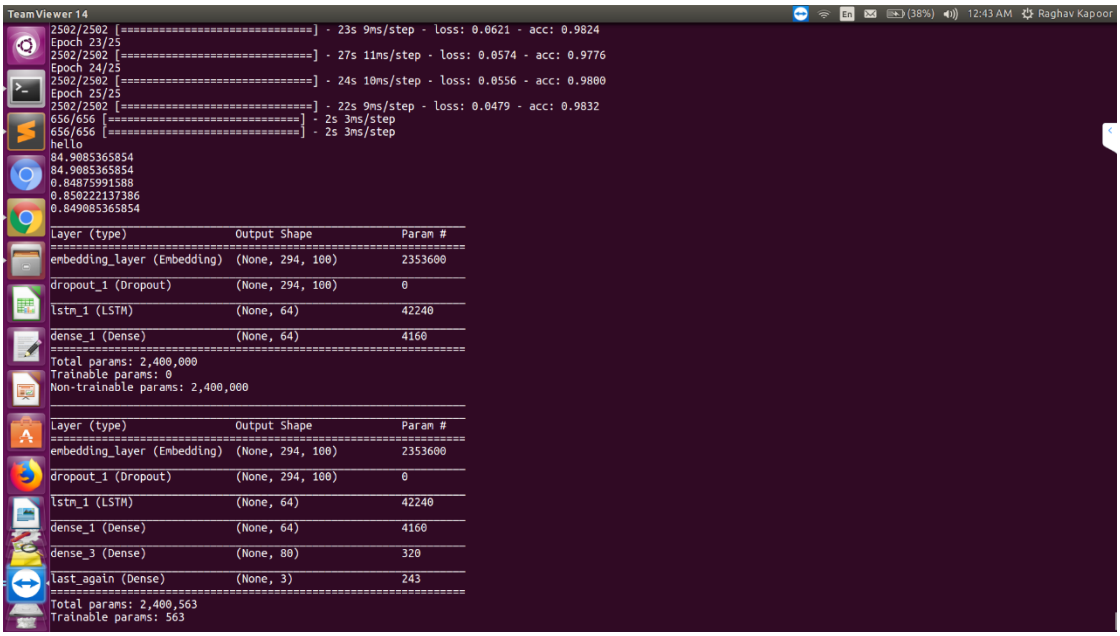


Figure 48 Running Model

6.6 Error Analysis

We analyze the possible reasons due to which our model gives error in its judgement.

(i) **OCR error:** We have used ocr.space for extracting text out of the memes. The memes in which the text is written in very small font, or the text written is slightly blurred, the OCR fails to recognize that text with 100% accuracy. Also, in many cases it is hard for the OCR reader to extract text which is written vertically or diagonally.

(ii) **Unconventional words (code switched):** A little work is done in dealing with uncommon Hinglish words which may arise due to spelling variations, grammatical errors or mixing of some regional languages by the creators of the memes. For example the spelling variation resulting from a difference in the pronunciation of the words can create a new set of words which are not present in the dictionary itself.

(iii) **Disguised hate:** Some memes are designed so that it might seem to be satirical to the annotators but might actually be inducing hate towards an individual in a disguised fashion. Such memes would not be correctly classified by our model. For example, "Abbe oh, ma***sa jane vale" which translates to "Hey, religious school going person"

(iv) **Overfitting of data:** Due to the training using deep learning models and also the memes on social media being repetitive, there might be some overfitting of data. We have tried to avoid the problem of overfitting by using the dropout layers and the best set of hyperparameters. However, the problem might still be present and may cause variation in the results.

(v) Ambiguity in Tweets: Sometimes the text or the tweets are ambiguous in nature and can convey two different meanings. For example, "I hate you, love". This can be a sarcastic way of telling a person that I hate you. Or may be a person jokingly might have said to another person that he hates the other but actually loves him. Such ambiguous statements are hard to detect and therefore can produce slight decline in the accuracy.

7. Applications

1. **Detect False Propaganda by Political Groups in Elections –** The illicit language used by various people can be detected through the system. These will ensure the right to speech to be exercised in a fruitful manner by all citizens and a clean election propaganda by the political groups.

2. **Youtube/Netflix Subtitles –** "Auto-beep" offensive language and whenever there is a word or phrase or a sentence which may hurt the feelings of some person or group, it will automatically be removed, thereby producing clean audio and subtitles.

3. **Online Social Media –** Can be used to report defamatory pages and comments on the social media platforms like Facebook, Twitter and Instagram. Many pages pop up during the election times and post material and videos and pictures which are not suitable for the internet. These can now be easily detected and removed.

4. **Feedback analytics for better user experience –** User feedbacks and the sentiments behind their comments can be evaluated. These analytics can be used to improve the product or improve customer service.

5. **Real time "clean-chat" facility –** Applications like Whatsapp and other chatbots could deploy the system which will help them to filter out the abusive and derogatory messages and the chats thus produced will be much better without the filtering system.

6. **Use by Censor boards –** It is often a practice by film makers to create content which might contain offensive language. Censor boards help review the content and remove if they find such cases. Currently this is done manually wherein a person watches the complete movie to eliminate that content. With the use of this

system the censor boards can auto-eliminate abusive and hate inducing content from the films.

8. Conclusion and Future Work

The problem of hate speech detection had started off as a child problem to sentiment analysis but in today's world, it has become a standalone problem and is in itself, a completely new research domain. As the problem of hate speech in code switched languages keeps on rising with the increasing reach of the internet, the need of the hour is to filter out all the illicit content on the social media so as to make it a cleaner environment for everyone. The task of handling code mixed data is a complex problem in itself involving great deal of intricacies in terms of varying semantics that code mixed language offers.

Through this work, we presented a pipeline which given Hinglish text can classify it into three categories: offensive, abusive and benign. This LSTM based model performed better than the other systems present for detecting hate speech in Hinglish language. We also introduced a novel dataset, i.e. the IPM (Indian Political Memes) Dataset which consists of images (Memes) classified in three categories - Benign, Satirical and Hate Inducing. Also, we proposed a pipeline to detect offense and hate from images which contain text in code switched languages. We developed a multi-channel CNN-LSTM model, which processes the images and text individually and combines the analysis from both channels to give the final classification result. The model plucks out the text from images and converts the text into a word vector representation before passing through the LSTM channel of the model. A number of different words embedding models were tried to attain the most optimum results. On the other hand, the images are passed through CNN channel of the model. We compare the results of our model to other deep learning based models and some supervised machine learning models, namely SVM and Random Forest classifier after extracting features from the images. The results suggest that our model outperforms all the other models producing state of the art results for Hinglish Language Memes classification on IPM dataset. These results also mark the fact that parallel analysis of text and image gives much better outcomes as compared to processing of images alone.

This work can be further extended to various avenues like inclusion of videos as a part of our analysis. A political speech video can be categorized as offensive by analysing the speech and the video graphics inspired by the work of [49]. The existing methods can also be used by various social media platforms to classify the complete page or user as objectionable. The pages on Facebook, or Twitter users who pop up during election times to provoke the masses can be detected and removed. Other GRU based models [50] can also be applied to this problem. Additionally, we can use our model to other code-switched language pairs. For example, the work by [51] detects sentiments from Chinese-English pair of code switched language. Also, the relative positions of words play a major role in the analysis of Hinglish text. So, we would like to explore such possibilities in our

future work [52]. We can also consider building a hate inducing video segmentation system [53] in order to remove hate inducing videos from the internet.

Appendix A

Tools Used

This appendix aims to provide a deep knowledge of the tools used to create a working code of the algorithms proposed by the authors. This section tries to provide an overview to the readers about the programming languages and libraries used to build the project.

A.1 PYTHON

Python is a standard high-level, general-purpose, easily understood, dynamically typed language. Code readability is one of the most important features of Python programming language. Python's syntax is such that it permits the programmers to articulate their concepts in much lesser lines of code as compared to other programming languages such as Java or C++. Python constructs intends to enable clear writing and easy to read programs, even though its length may be large.

Many programming paradigms are supported like object-oriented, imperative and purposeful programming, or procedural programming style. Also, it gives a dynamic typed system, along with a large and comprehensive standard libraries making programming easy for the beginners as well. Python compilers are available for almost all the popular operating systems, enabling it to run on a wide range of systems.

A.1.1 Numpy and Pandas

NumPy is a famous open source Python package developed for processing general purpose arrays. NumPy package is used for expeditiously manipulating massive multi-dimensional arrays while not sacrificing an excessive amount of speed for tiny multi-dimensional arrays. NumPy has ability for discrete Fourier transform, random number generation and basic linear algebra. It is the elementary package for scientific calculations in Python. NumPy contains a robust N-dimensional array object, refined (broadcasting) functions; integration tools, C/C++ and algebraic language codes, helpful algebra, Fourier rework, and skills for random number generation. Also, it can even be used as an effective multi-dimensional container of generic information. NumPy also provides facility for naming arbitrary data types which helps it to efficiently integrate with a large number of databases.

Pandas is also an open source library which provides superior, easy-to-use information structures and information analysis tools for the Python language. Pandas has various

features like Data Frame object for data manipulation with integrated compartmentalization, modules for reading and writing information in different data structures and file formats.

Code snippets for data manipulation using Pandas are presented below:

```
def load_df(dataset_name):
    """
    Load data as DataFrame for logistic regression and ensemble models, without sampling
    :param dataset_name: name of dataset
    :return: DataFrame
    """
    print('Loading dataset DataFrame')
    if dataset_name == 'EP' or dataset_name == 'ep':
        file_name = './data/experience_project/ep.csv'
        df = pd.read_csv(file_name)
        return df
    elif dataset_name == 'twitter' or dataset_name == 'Twitter':
        file_name = './data/twitter/TD.csv'
        df = pd.read_csv(file_name)
        df = df.dropna()
        return df
    elif dataset_name in ['popular', 'all', 'AskReddit', 'books', 'gaming', 'movies', 'Jokes']:
        file_name = './data/reddit/{}.csv'.format(dataset_name)
        df = pd.read_csv(file_name)
        return df
    else:
        raise ValueError("Error: unrecognized dataset")


def load_data(dataset_name):
    """
    Load data for NN models, with under sampling from Twitter dataset
    :param dataset_name: name of dataset
    :return: X, y
    """
    print('Loading text dataset')
    if dataset_name == 'EP' or dataset_name == 'ep':
        file_name = './data/experience_project/ep.csv'
        df = pd.read_csv(file_name)
        X = df['body'].as_matrix()
        y = df['y'].as_matrix()
        return X, y
    elif dataset_name == 'twitter' or dataset_name == 'Twitter':
        file_name = './data/twitter/TD.csv'
        df = pd.read_csv(file_name)
        df = df.dropna()
        print(df.describe)
        df_pos = df.loc[df['y'] == 1]
        df_neg = df.loc[df['y'] == 0]
        df = pd.concat([df_pos, df_neg.sample(len(df_pos['y']))])
        df = df.sample(frac=1)
        X = df['tweet'].as_matrix()
        y = df['y'].as_matrix()
        return X, y
    elif dataset_name in ['popular', 'all', 'AskReddit', 'books', 'gaming', 'movies', 'Jokes']:
        file_name = './data/reddit/{}.csv'.format(dataset_name)
        df = pd.read_csv(file_name)
        X1 = df['title'].as_matrix()
        X2 = df['usertext'].as_matrix()
        y = df['y'].as_matrix()
        return X1, X2, y
    else:
        raise ValueError("Error: unrecognized dataset")
```

A.1.2 Matplotlib

Matplotlib is a 2D plotting library for Python which helps us to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib may be utilized in Python scripts, the jupyter notebook, net application servers, the Python and IPython shell and for graphical program toolkits. Matplotlib provides features to develop plots, histograms, bar charts, error charts, power spectra, etc., with a very few lines of codes.

```
import numpy as np
import matplotlib.pyplot as plt

# Compute the x and y coordinates for points on a sine curve
x = np.arange(0, 3 * np.pi, 0.1)
y = np.sin(x)
plt.title("sine wave form")

# Plot the points using matplotlib
plt.plot(x, y)
plt.show()
```

A.1.3 Scikit-learn

Scikit-learn is an open source machine learning module available in the Python programming language. There are many features of Scikit-learn like classification, regression and agglomeration algorithms together with support vector machines, gradient boosting, random forests, *k*-means clustering, and is designed to consistent with the Python libraries like NumPy and SciPy.

A code snippet using Scikit-Learn is shown below:

```
def evaluate_prediction(y_test, y_pred, k_th, model_name, dataset_name):
    fpr, tpr, th = metrics.roc_curve(y_test, y_pred)
    roc_auc = metrics.auc(fpr, tpr)

    df_results = pd.DataFrame({'y_test': y_test, 'y_pred': y_pred}, index=None)
    df_results.to_csv('./save/prediction_{}_{}_{}.csv'.format(dataset_name, model_name, k_th), index=False)

    o_threshold = 0.5
    for i in range(len(y_pred)):
        if y_pred[i] >= o_threshold:
            y_pred[i] = 1
        else:
            y_pred[i] = 0

    acc = metrics.accuracy_score(y_test, y_pred)
    pre = metrics.precision_score(y_test, y_pred)
    rec = metrics.recall_score(y_test, y_pred)
    f1 = metrics.f1_score(y_test, y_pred)

    dict_eval = {'date': datetime.date.today(),
                 'model': model_name,
                 'accuracy': acc,
                 'precision': pre,
                 'recall': rec,
                 'f-score': f1,
                 'roc': roc_auc,
                 'note': '{}_th fold'.format(k_th),
                 'dataset': dataset_name
                 }
    with open('./output/{}.csv'.format(dataset_name), 'a') as f:
        field_names = ['date', 'model', 'accuracy', 'precision', 'recall', 'f-score', 'roc', 'note', 'dataset']
        writer = csv.DictWriter(f, fieldnames=field_names)
        writer.writerow(dict_eval)
    return acc, pre, rec, f1, roc_auc
```

```
    # 10-fold cross validation
    num_fold = 10
    kf = KFold(n_splits=num_fold, shuffle=True, random_state=0)
    for train_index, test_index in kf.split(X):
        num_fold -= 1
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y[train_index], y[test_index]
        # Logisitc Regression
        clf = LogisticRegression(penalty='l2', tol=1e-6)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test)[:,1]
        acc, pre, rec, f1, auc = evaluate_prediction(y_test, y_pred, k_th=num_fold,
                                                     model_name='Logistic Regression', dataset_name=args.dataset)
        lr_acc.append(acc)
        lr_pre.append(pre)
        lr_rec.append(rec)
        lr_f1.append(f1)
        lr_auc.append(auc)
        # Random Forest
        clf = RandomForestClassifier(n_estimators=20, max_depth=8, random_state=0)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test)[:, 1]
        acc, pre, rec, f1, auc = evaluate_prediction(y_test, y_pred, k_th=num_fold,
                                                     model_name='Random Forest', dataset_name=args.dataset)
        rf_acc.append(acc)
        rf_pre.append(pre)
        rf_rec.append(rec)
        rf_f1.append(f1)
        rf_auc.append(auc)
        # GBDT
        clf = GradientBoostingClassifier(max_depth=8, random_state=0)
        clf.fit(X_train, y_train)
        y_pred = clf.predict_proba(X_test)[:, 1]
        acc, pre, rec, f1, auc = evaluate_prediction(y_test, y_pred, k_th=num_fold,
                                                     model_name='GBDT', dataset_name=args.dataset)
        gbdt_acc.append(acc)
        gbdt_pre.append(pre)
        gbdt_rec.append(rec)
        gbdt_f1.append(f1)
        gbdt_auc.append(auc)
        # XGBoost
        dtrain = xgb.DMatrix(X_train, label=y_train, missing=-999)
        dtest = xgb.DMatrix(X_test, label=y_test, missing=-999)
        params = {'max_depth': 10, 'eta': 0.1, 'silent': 1, 'objective': 'binary:logistic', 'nthread': -1}
        num_round = 10000
        watchlist = [(dtrain, 'train'), (dtest, 'test')]
        model = xgb.train(params, dtrain, num_round, watchlist, early_stopping_rounds=50, verbose_eval=10)
        y_pred = model.predict(dtest)
        acc, pre, rec, f1, auc = evaluate_prediction(y_test, y_pred, k_th=num_fold,
                                                     model_name='XGBoost', dataset_name=args.dataset)
        xgb_acc.append(acc)
        xgb_pre.append(pre)
        xgb_rec.append(rec)
        xgb_f1.append(f1)
        xgb_auc.append(auc)
result_average.append(['Logistic Regression', np.mean(lr_acc), np.mean(lr_pre), np.mean(lr_rec), np.mean(lr_f1),
    np.mean(lr_auc)])
result_average.append(['Random Forest', np.mean(rf_acc), np.mean(rf_pre), np.mean(rf_rec), np.mean(rf_f1),
    np.mean(rf_auc)])
result_average.append(['GBDT', np.mean(gbdt_acc), np.mean(gbdt_pre), np.mean(gbdt_rec), np.mean(gbdt_f1),
    np.mean(gbdt_auc)])
result_average.append(['XGB', np.mean(xgb_acc), np.mean(xgb_pre), np.mean(xgb_rec), np.mean(xgb_f1), np.mean(xgb_auc)])
print(tabulate(result_average, headers=h))
```

A.1.4 Keras

Keras is an API for high-level neural networks, written in Python which can run on Tensor Flow, Theano and CNTK. Keras was developed with the aim of enabling quick

experimentation. Keras is compatible with: Python 2.7-3.6. The best style of model is that the sequential model, a linear stack of layers. All Keras layers have a number of methods in common like,

(i) e.layer.get_weights ()

(ii) f.layer.set_weights (weights)

(iii) a.layer.get_config ().

Code snippets of the model architectures are presented below:

```
from keras import layers

config = layer.get_config()
layer = layers.deserialize({'class_name': layer.__class__.__name__,
                            'config': config})
```

```
def build_RNN(args):
    inp = Input(shape=(args.max_seq_len, ))
    x = Embedding(args.max_num_words, args.embedding_dim)(inp)
    x = SimpleRNN(args.rnn_units, return_sequences=True)(x)
    x = GlobalMaxPool1D()(x)
    x = Dropout(rate=args.dropout_rate)(x)
    x = BatchNormalization()(x)
    x = Dense(units=args.dense_units, activation="relu")(x)
    x = Dropout(rate=args.dropout_rate)(x)
    x = BatchNormalization()(x)
    x = Dense(1, activation="sigmoid")(x)
    model = Model(inputs=inp, outputs=x)
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

```
def build_LSTM(args):
    inp = Input(shape=(args.max_seq_len, ))
    x = Embedding(input_dim=args.max_num_words, output_dim=args.embedding_dim)(inp)
    x = LSTM(units=args.lstm_units, activation='tanh', dropout=args.dropout_rate, return_sequences=True)(x)
    x = GlobalMaxPool1D()(x)
    x = Dropout(args.dropout_rate)(x)
    x = BatchNormalization()(x)
    x = Dense(args.dense_units, activation="relu")(x)
    x = Dropout(args.dropout_rate)(x)
    x = BatchNormalization()(x)
    x = Dense(1, activation="sigmoid")(x)
    model = Model(inputs=inp, outputs=x)
    model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

A.2 Twitter API

The idea behind TwitterAPI's design is to give a single method of accessing the various tweets from a particular twitter handle on Twitter. We just need to call the request method with any endpoint found on Twitter's developer site, and all the desired tweets would be scrapped. There are various advantages of using this APl like less code to maintain, and just a single method to learn.

Here is a code snippet of a API request:

```
from TwitterAPI import TwitterAPI
api = TwitterAPI(consumer_key, consumer_secret, access_token_key, access_token_secret)
r = api.request('search/tweets', {'q':'pizza'})
print r.status_code
```

The following code snippet gives entire Twitter's response as one long string.

```
for item in r.get_iterator():
    print item['user']['screen_name'], item['text']
```

A.3 Google_image_download

Google_image_download is a python Script that can be used for searching and downloading thousands of images from Google directly.

The arguments can be passed either directly from the command as in the examples shown below or it can be passed through a config file. Below is a snippet showing how a config file looks.

More than one record can be passed through a config file. The below code snippet contains two series of records. This code will run through each of the record iteratively and the images would be downloaded based on arguments passed.

```
from google_images_download import google_images_download   #importing the library

response = google_images_download.googleimagesdownload()   #class instantiation

arguments = {"keywords":"Polar bears,baloons,Beaches","limit":20,"print_urls":True}   #creating list
paths = response.download(arguments)   #passing the arguments to the function
print(paths)   #printing absolute paths of the downloaded images
```

A.4 OCR.SPACE

The open source OCR API (*ocr.space*) provides an easy method of parsing pictures and multi-page PDF documents (PDF OCR) and obtaining the extracted text results in a JSON format. This API can be used from any internet-connected device like desktop, android etc. The results are returned by the API in a JSON format. The results usually contain the Exit Code, Error details and the parsed results for the Image or PDF pages that are input to the system.

Here is an example of how to access the API from Python using the request.post command.

```
18      payload = {'isOverlayRequired': overlay,
19                 'apikey': api_key,
20                 'language': language,
21                 }
22      with open(filename, 'rb') as f:
23          r = requests.post('https://api.ocr.space/parse/image',
24                            files={filename: f},
25                            data=payload,
26                            )
```

References

[1] K. Bali, J. Sharma, M. Choudhury, and Y. Vyas, "' I am borrowing ya mixing?' An Analysis of English-Hindi Code Mixing in Facebook," in *Proceedings of the First Workshop on Computational Approaches to Code Switching*, 2014, pp. 116–126.

[2] A. Das and B. Gambäck, "Identifying languages at the word level in code-mixed indian social media text," in *Proceedings of the 11th International Conference on Natural Language Processing*, 2014, pp. 378–387.

[3] D. Mave, S. Maharjan, and T. Solorio, "Language Identification and Analysis of Code-Switched Social Media Text," in *Proceedings of the Third Workshop on Computational Approaches to Linguistic Code-Switching*, 2018, pp. 51–61.

[4] K. Singh, I. Sen, and P. Kumaraguru, "Language identification and named entity recognition in hinglish code mixed tweets," in *Proceedings of ACL 2018, Student Research Workshop*, 2018, pp. 52–58.

[5] Y. Vyas, S. Gella, J. Sharma, K. Bali, and M. Choudhury, "Pos tagging of english-hindi code-mixed social media content," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 974–979.

[6] D. Gupta, S. Tripathi, A. Ekbal, and P. Bhattacharyya, "SMPOST: parts of speech tagger for code-mixed Indic social media text," *arXiv preprint arXiv:1702. 00167*, 2017.

[7] P. R. K. Rao and S. L. Devi, "CMEE-IL: Code Mix Entity Extraction in Indian Languages from Social Media Text@ FIRE 2016-An Overview," in *FIRE (Working Notes)*, 2016, pp. 289–295.

[8] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," 2001.

[9] A. Prabhu, A. Joshi, M. Shrivastava, and V. Varma, "Towards sub-word level compositions for sentiment analysis of hindi-english code mixed text," *arXiv preprint arXiv:1611. 00472*, 2016.

[10] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, 2012, pp. 90–94.

[11] B. Pang, L. Lee, and Others, "Opinion mining and sentiment analysis," *Foundations and Trends® in Information Retrieval*, vol. 2, no. 1–2, pp. 1–135, 2008.

[12] S. Sharma, P. Srinivas, and R. C. Balabantaray, "Text normalization of code mix and sentiment analysis," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2015, pp. 1468–1473.

[13] M. G. Jhanwar and A. Das, “An Ensemble Model for Sentiment Analysis of Hindi-English Code-Mixed Data,” *arXiv preprint arXiv:1806.04450*, 2018.

[14] D. Gupta, A. Lamba, A. Ekbal, and P. Bhattacharyya, “Opinion Mining in a Code-Mixed Environment: A Case Study with Government Portals,” in *Proceedings of the 13th International Conference on Natural Language Processing*, 2016, pp. 249–258.

[15] E. Spertus, “Smokey: Automatic recognition of hostile messages,” in *Aaai/iaai*, 1997, pp. 1058–1065.

[16] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, “Detection of harassment on web 2.0,” *Proceedings of the Content Analysis in the WEB*, vol. 2, pp. 1–7, 2009.

[17] S. O. Sood, E. F. Churchill, and J. Antin, “Automatic identification of personal insults on social news sites,” *J. Am. Soc. Inf. Sci. Technol.*, vol. 63, no. 2, pp. 270–285, 2012.

[18] G. Xiang, B. Fan, L. Wang, J. Hong, and C. Rose, “Detecting offensive tweets via topical feature discovery over a large scale twitter corpus,” in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 1980–1984.

[19] F. Del Vigna12, A. Cimino23, F. Dell’Orletta, M. Petrocchi, and M. Tesconi, “Hate me, hate me not: Hate speech detection on Facebook,” 2017.

[20] A. Bohra, D. Vijay, V. Singh, S. S. Akhtar, and M. Shrivastava, “A Dataset of Hindi-English Code-Mixed Social Media Text for Hate Speech Detection,” in *Proceedings of the Second Workshop on Computational Modeling of People’s Opinions, Personality, and Emotions in Social Media*, 2018, pp. 36–41.

[21] P. Mathur, R. Shah, R. Sawhney, and D. Mahata, “Detecting offensive tweets in hindi-english code-switched language,” in *Proceedings of the Sixth International Workshop on Natural Language Processing for Social Media*, 2018, pp. 18–26.

[22] P. Mathur, R. Sawhney, M. Ayyar, and R. Shah, “Did you offend me? classification of offensive tweets in hinglish language,” in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, 2018, pp. 138–148.

[23] W. Wei-ning, Y. Ying-lin, and J. Sheng-ming, “Image retrieval by emotional semantics: A study of emotional space and feature extraction,” in *2006 IEEE International Conference on Systems, Man and Cybernetics*, 2006, vol. 4, pp. 3534–3539.

[24] S. Siersdorfer, E. Minack, F. Deng, and J. Hare, “Analyzing and predicting sentiment of images on the social web,” in *Proceedings of the 18th ACM international conference on Multimedia*, 2010, pp. 715–718.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[26] Q. You, J. Luo, H. Jin, and J. Yang, “Robust image sentiment analysis using progressively trained and domain transferred deep networks,” in *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[27] G. Cai and B. Xia, “Convolutional neural networks for multimedia sentiment analysis,” in *Natural Language Processing and Chinese Computing*, Springer, 2015, pp. 159–167.

[28] S. Poria, I. Chaturvedi, E. Cambria, and A. Hussain, “Convolutional MKL based multimodal emotion recognition and sentiment analysis,” in *2016 IEEE 16th international conference on data mining (ICDM)*, 2016, pp. 439–448.

[29] L.-P. Morency, R. Mihalcea, and P. Doshi, “Towards multimodal sentiment analysis: Harvesting opinions from the web,” in *Proceedings of the 13th international conference on multimodal interfaces*, 2011, pp. 169–176.

[30] R. Kapoor, Y. Kumar, K. Rajput, R. R. Shah, P. Kumaraguru, and R. Zimmermann, “Mind Your Language: Abuse and Offense Detection for Code-Switched Languages,” *arXiv preprint arXiv:1809. 08652*, 2018.

[31] T. Davidson, D. Warmsley, M. Macy, and I. Weber, “Automated hate speech detection and the problem of offensive language,” in *Eleventh International AAAI Conference on Web and Social Media*, 2017.

[32] J. Cohen, “A coefficient of agreement for nominal scales,” *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[33] R. Barnett *et al.*, “The LIDES Coding Manual: A Document for Preparing and Analyzing Language Interaction Data Version 1.1--July 1999,” *International Journal of Bilingualism*, vol. 4, no. 2, pp. 131–271, 2000.

[34] J. L. Fleiss, “Measuring nominal scale agreement among many raters,” *Psychol. Bull.*, vol. 76, no. 5, p. 378, 1971.

[35] J. Pennington, R. Socher, and C. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[36] F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, “Multimedia Lab @ ACL WNUT NER Shared Task: Named Entity Recognition for Twitter Microposts using Distributed Word Representations,” in *Proceedings of the Workshop on Noisy User-generated Text*, 2015, pp. 146–153.

[37] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. “ O’Reilly Media, Inc.,” 2008.

[38] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, 1999, vol. 2, pp. 246–252.

[39] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[41] D. A. Van Dyk and X.-L. Meng, “The art of data augmentation,” *J. Comput. Graph. Stat.*, vol. 10, no. 1, pp. 1–50, 2001.

[42] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[43] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[44] S. J. Pan and Q. Yang, “A survey on transfer learning,” *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, 2009.

[45] R. M. Haralick and L. G. Shapiro, *Computer and robot vision*, vol. 1. Addison-wesley Reading, 1992.

[46] R. Datta, D. Joshi, J. Li, and J. Z. Wang, “Studying aesthetics in photographic images using a computational approach,” in *European conference on computer vision*, 2006, pp. 288–301.

[47] H. Tamura, S. Mori, and T. Yamawaki, “Textural features corresponding to visual perception,” *IEEE Trans. Syst. Man Cybern.*, vol. 8, no. 6, pp. 460–473, 1978.

[48] P. Viola and M. J. Jones, “Robust real-time face detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[49] V. P. Rosas, R. Mihalcea, and L.-P. Morency, “Multimodal sentiment analysis of spanish online videos,” *IEEE Intell. Syst.*, vol. 28, no. 3, pp. 38–45, 2013.

[50] Z. Zhang, D. Robinson, and J. Tepper, “Detecting hate speech on twitter using a convolution-gru based deep neural network,” in *European Semantic Web Conference*, 2018, pp. 745–760.

[51] Z. Wang, S. Lee, S. Li, and G. Zhou, “Emotion detection in code-switching texts via bilingual and sentimental information,” in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, 2015, vol. 2, pp. 763–768.

[52] A. D. Shaikh, M. Jain, M. Rawat, R. R. Shah, and M. Kumar, “Improving accuracy of sms based faq retrieval system,” in *Multilingual Information Access in South Asian Languages*, Springer, 2013, pp. 142–156.

[53] R. R. Shah, Y. Yu, A. D. Shaikh, S. Tang, and R. Zimmermann, “ATLAS: automatic temporal segmentation and annotation of lecture videos based on modelling transition time,” in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014, pp. 209–212.

Publication Status

We have written the following research papers:

1. Title: "Mind Your Language: Abuse and Offense Detection for Code-Switched Languages"

Accepted in:

33rd AAAI-2019 Conference (Association for the Advancement of Artificial Intelligence), held in Honolulu, Hawaii, U.S.A. from 27 January, 2019 to 1 February, 2019.

h5-index: 69

h5-media: 101

2. Title: "Hate Me Not: Detecting Hate Inducing Memes in Code Switched Languages"

Submitted in:

28th ACM International Conference on Information and Knowledge Management (CIKM), held in Beijing, China, from November 3rd-7th, 2019

h5-index: 49

h5-median: 64